Titre: Title:	Navigation semi-autonome d'un fauteuil roulant motorisé dans un environnement extérieur par intégration d'un GPS
Auteur: Author:	Bassel Shaer
Date:	2011
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Shaer, B. (2011). Navigation semi-autonome d'un fauteuil roulant motorisé dans un environnement extérieur par intégration d'un GPS [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/519/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/519/
Directeurs de recherche: Advisors:	Richard Gourdeau
Programme: Program:	Génie électrique

UNIVERSITÉ DE MONTRÉAL

NAVIGATION SEMI-AUTONOME D'UN FAUTEUIL ROULANT MOTORISÉ DANS UN ENVIRONNEMENT EXTÉRIEUR PAR INTÉGRATION D' UN GPS

BASSEL SHAER
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)
MARS 2011

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

NAVIGATION SEMI-AUTONOME D'UN FAUTEUIL ROULANT MOTORISÉ DANS UN ENVIRONNEMENT EXTÉRIEUR PAR INTÉGRATION D' UN GPS

présenté par : SHAER, Bassel
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. HURTEAU, Richard, D.Ing., président.
M. GOURDEAU, Richard, Ph.D., membre et directeur de recherche.
M. COHEN, Paul, Ph.D., membre.

Pour les recherches déstinées aux fins humanitaires
pour les personnes handicapées . . .

REMERCIEMENTS

Je tiens à remercier mon directeur de recherche Monsieur Richard Gourdeau pour m'avoir dirigé ma maîtrise. Je remercie Monsieur Paul Cohen pour m'avoir accueilli dans le lab Groupe de Recherche en Perception et Robotique (GRPR). Je remercie fortement aussi Monsieur Sousso Kelouwani pour m'avoir aidé directement dans mes travaux de maîtrise.Je remercie aussi tous les membres du GRPR pour leur aide. Je remercie ma famille, grace à leur soutien, j'ai réussi.

RÉSUMÉ

La navigation à l'extérieur ou dans un milieu urbain représente un défi important dans la navigation robotique. Pour y parvenir, le robot doit pouvoir se localiser, planifier sa trajectoire et éviter les obstacles. Ce travail présente une étude préliminaire de ces différents aspects dans la perspective d'une application éventuelle à des fauteuils roulants motorisés.

Pour la localisation en milieu urbain, l'utilisation du capteur GPS commercial a été privilégiée. En premier lieu, une caractérisation des erreurs de mesures GPS a été réalisée. Ces erreurs comportent deux composantes : un biais qui varie lentement et une erreur aléatoire. Deux appareils GPS ont été utilisés : le premier pour déterminer le trajet désiré en extrayant les données à partir d'un fichier GPX généré par l'appareil suite à une planification en mode piéton ; le deuxième est un senseur GPS qui donne la position instantanée du robot sous forme de phrases suivant le protocole NMEA. Un algorithme de projection des données GPS dans le repère local en fonction de la position initiale spécifiée par l'usager permet de compenser, à court terme, le biais des mesures GPS.

Une fusion de données GPS et des mesures d'odométrie fournies par le robot a été réalisée à l'aide d'un filtre de Kalman étendu. Les résultats montrent que le robot peut suivre des trajectoires planifiées avec une précision de l'ordre de 0,5 mètre. Un suivi de trajectoire rectiligne le long d'un mur a aussi été réalisé combinant le suivi de trajectoire avec la localisation et l'évitement d'un obstacle utilisant un capteur laser pour détecter la « bordure » du chemin (de façon analogue au déplacement sur un trottoir).

Ces résultats préliminaires illustrent le potentiel de l'approche proposée et ouvrent la voie à des travaux plus élaborés traitant des différentes problématiques de la navigation en milieu urbain : localisation, planification et évitement d'obstacles.

ABSTRACT

Navigation in exterior or urban environments is an challenging problem in a robotic navigation. In order to achieve this goal, the robot must be able to localize itself, carry out path planing and avoid obstacles. This work presents a preliminary study of these various aspects in anticipation of a possible application to motorized wheelchairs.

For localization in urban areas, the use of commercial GPS sensors has been privileged. In first place, a characterization of GPS measurement errors was performed. These errors have two components: a bias that varies slowly and a random error. Two devices GPS were used: one to determine the desired path by extracting data from a file GPX, which is generated by the navigator GPS using the settings for pedestrian mode; the other one is a sensor GPS which gives the instantaneous position of robot in the form of sentences according to the NMEA protocol. A projection algorithm of GPS data in the local coordinate plane with respect to the initial position specified by the user in order to compensate, for short term, the measurements bias of the GPS. Data fusion of GPS data and odometry measurements was performed using an extended Kalman filter. The results show that the robot can follow the trajectories planned with an accuracy of about 0.5 meters.

Trajectory tracking along a straight wall has also been done combining the trajectory tracking with the localization and avoidance of obstacles using a laser sensor to detect the “edges” of the path (analogous to the movement on a sidewalk).

Theses preliminary results show the potential of the proposed approach and open the way for more extensive work dealing with various problems of navigation in urban areas: localization, planification and obstacle avoidance.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES ANNEXES

LISTE DES SIGLES ET ABRÉVIATIONS

GPS :	Global Positioning System.
PPS :	Precise Positioning System.
SPS :	Standard Postioning System.
WG84 :	World Geodetic System 1984.
WASS :	Wide Area Augmentation System.
NMEA :	National Marine Electronics Association .
XML :	Extensible Markup Language.
ECEF :	Earth-Centered, Earth-Fixed.
NavPointToPoint :	Navigation Point To Point.
Den1 :	dénivellation de droit.
Den2 :	dénivellation de gauche.
MRDD :	marge de distance à droit.
MRDG :	marge de distance à gauche.
OdoReceiver :	OdométrieReceiver.
EKF :	extended Kalman filter.

CHAPITRE 1

REVUE BIBLIOGRAPHIQUE

1.1 Introduction

Dans la navigation robotique, la localisation du robot est un élément essentiel et l'objectif d'un grand nombre de projets de recherche. Les méthodes pour localiser un robot peuvent utiliser des cameras, des lasers, les GPS et des cartes, des balises, etc (Jarvis, 2010). La difficulté de la localisation dépend de l'environnent, des appareils utilisés et de la tâche à réaliser. Les applications qui demandent une bonne localisation sont nombreuses quelles soient militaires, industrielles ou civiles. La détection d'obstacles est le deuxième élément à mettre en oeuvre après la localisation (Laugier *et al*., 2001). L'objectif de la détection des obstacles est soit de changer la trajectoire planifiée, soit de faire un évitement réactif de collision. La nécessité de reconstruire un modèle de l'environnement implique bien souvent plus d'une source d'informations à partir des quelles une fusion de données sera faite pour estimer la position d'un obstacle présumé. Les études faites dans ce domaine font souvent l'hypothèse d'un environnement plat pour y détecter les dénivellations et les obstacles potentiels (Chang *et al*., 1999). Dans un système complet de robotique mobile, la détection des obstacles n'est qu'une des tâches à réaliser et est liée étroitement avec les tâches de localisation et d'évitement d'obstacles. Ainsi, lorsque l'on parle d'une détection d'obstacle, cela implique l'évitement et les manoeuvres qui garantissent un mouvement sécuritaire. Dans ce chapitre, on recense certains travaux développés pour réaliser des tâches ressemblant à celles qui seront développée dans les chapitres suivants. Les approches discutées ci-dessous présentent des solutions à des problèmes similaires mais pour des contextes d'applications différents.

1.2 Méthodes basées sur l'utilisation de laser incliné avec cameras

Les cameras sont des outils intéressants pour aider le robot à savoir où il est en extrayant les données et les dimensions à partir des images acquises pour calculer la position de robot. Donc, la camera est considérée comme une première source de données sur laquelle des algorithmes seront appliqués pour déterminer les repères, la position de robot et trouver des objets précis dans une image (Cobos *et al*., 2010). Le laser vient au second lieu pour aider ces algorithmes à améliorer leurs précisions et donner des mesures brutes et justes dans une forme particulière (Vincent *et al*., 2010).

Les algorithmes d'analyse d'images, pour détecter les dénivellations et les frontières d'un trottoir,

basés sur le gradient appliquent un seuil sur l'amplitude du gradient d'une image Wijesoma *et al*. (2002),Li *et al*. (2010). La performance des méthodes de gradient est considérée bonne si l'image a des régions uniformes avec une bonne séparation entre les régions. En environnements extérieurs, les conditions climatiques font que, souvent, l'éclairage ne donnent pas une image bien contrastée. C'est pour cela que Wijesoma *et al*. (2002) ont suggéré une méthode basée sur la vision et le laser. Le laser peut fournir des données justes dans de mauvaises conditions climatiques (d'éclairage), mais son utilisation est limitée à la détection des obstacles. Alors, la projection de données laser avec une image peut aider à bien localiser le robot. Pour y parvenir, l'EKF (Extended Kalman filter) a été utilisé pour déterminer le milieu de la rue en se basant sur la détection des trottoirs. L'appareil laser est incliné d'un certain angle afin d'effectuer un scan devant le robot permettant ainsi de construire un vecteur discontinu ou formé de plusieurs segments. Il est supposé que la rue et le trottoir sont des plans et horizontaux. Le plan vertical représente alors une dénivellation. Lorsque le vecteur de mesures est construit, l'algorithme développé a pour rôle de distinguer l'intersection entre les différentes plans : le plan du laser et le plan de surface de la route. Alors l'intersection correspond à une discontinuité significative dans cette ligne de données ou bien un changement de gradient dans l'amplitude des données. Le bruit de mesure peut corrompre les points en rendant la détection des points de discontinuité difficile. Le problème de filtrage de bruit de mesures et détection de points de discontinué a été traité par l'EKF, donc pour un ensemble de données d_i fourni par le laser. Le modèle est fait de sorte à prédire les données suivantes d_{i+1} connaissant les deux mesures précédentes (d_{i-1},d_i) sous l'hypothèse que les points évoluent selon une ligne droite. Ce modèle de processus combiné avec l'ensemble de données utilise le filtre de Kalman pour obtenir un estimé filtré d'un ensemble de mesures. Le filtrage sera efficace et valide si le modèle décrit l'évolution de l'ensemble des points de données. Cela est exact pour des donnés dans la même région. Étant donné que les paramètres de processus pour les deux régions sont différents, la prédiction sera mauvaise pour les mesures sur les bords séparant les deux régions bien que la structure de modèle soit la même. L'erreur de prédiction est proprement référencée pour obtenir des points candidats des discontinuités ou des dénivellations.

Truong *et al*. (2007) ont suggéré une méthode pour aider les soldats à éviter les tonneaux lors de la conduite de véhicules militaires. Le problème est que le conducteur d'une voiture ne peut pas être en mesure de déterminer les menaces de tonneaux comme les escarpements, le sol mou, l'eau ou les dalots situés dans un des côtés du véhicule. Pour réduire le nombre d'accidents causés par les tonneaux et détecter les dénivellations, un système d'alarme a été développé pour identifier les menaces de tonneaux et avertir le conducteur. Ce système utilise un algorithme de traitement de l'image basé sur la neurobiologie de la vision d'un insecte. Le système contient une camera, un appareil laser planaire avec une longue portée et un module de traitement dans lequel un processeur d'image biomimétique détecte en temps réel la dénivellation présente dans l'image et présente

une vision renforcée au conducteur (DVE : Driver's Vision Enhancer) affichant les frontières détectées dans l'image courante et les dénivellations abruptes de la rue. L'armée a développé cette technologie sur les voitures de type HMMWV(High Mobility Multipurpose Wheeled Vehicle) qui font beaucoup d'accidents en raison de la limitation du champ de vision. Le système contient une caméra vidéo monochrome, une caméra d'une grande longueur d'onde infrarouge (LWIR) et un télémètre laser.

L'algorithme développé prend des images à partir de la caméra et de la vidéo, les images ne sont pas bien orientées et conformes. Donc des traitements et rotation seront faits pour les superposer l'une sur l'autre et les orienter pour que le point de vue soit le même. L'algorithme fusionne les données et combine les dénivellations détectées par chacune des caméras en une seule gamme de dénivellations qui peuvent être affichées au conducteur. L'algorithme de fusion utilise une somme pondérée par l'intensité des pixels des dénivellations dans les deux images, intégrant ainsi les informations dans une seule image.

Un autre algorithme a été requis car d'autres objets, comme des arbres et des bâtiments, sont affichés dans l'image. Pour ne pas distraire le conducteur, les frontières de la rue doivent être bien définies. Une technique a été développée par (Wilson et Dickson, 1999) appelée POPPET. Cet algorithme fait l'interpolation entre les segments consécutifs des frontières de la rue.

Cet algorithme cherche l'origine des frontières principales dans le quart inférieur de droit de l'image en cherchant l'origine de la ligne la plus basse et la plus à droite. Ensuite un vecteur peut être désigné à partir de l'origine. En comparant le croisement avec un seuil, ce seuil est relié avec un temps de réaction de la part de conducteur. Un signal acoustique et visuel avertit simultanément le conducteur pour éviter le danger.

Chang *et al*. (1999) ont développé une méthode pour la navigation autonome sur les routes. La télédétection par laser de type *LADAR* (Laser Detection and Ranging) est utilisée pour détecter les obstacles et régions de dissimulation. Dans ce travail, la cartographie a été utilisée pour représenter les obstacles : négatifs, positifs et cachés. De plus l'algorithme de cartographie utilise aussi le GPS et un système de navigation inertielle INS. L'objectif du module de détection des obstacles est d'extraire des régions non navigables. Des méthodes sont proposées pour extraire les obstacles d'un ensemble d'images en utilisant la profondeur et la pente pour avoir un détecteur rapide.

L'image du LADAR contient 128 lignes de scan vertical, chacune contenant un ensemble de 64 données. Les données sont traitées pendant le processus de lecture permettant ainsi de détecter les obstacles dès que les données sont disponibles. Pour chaque ligne de scan, le filtre de kalman est appliqué aux données. Le filtre élimine les points aberrants causés par des réflexions spéculaires.

Après avoir filtré les données, les coordonnées cartésiennes de chaque pixel dans la ligne de scan sont calculées dans le repère du système qui est le centre du véhicule, les valeurs x,y,z (coordonnées cartésiennes) et la distance de proximité r assignée à chaque pixel sont utilisés pour

calculer les dérivées de la profondeur et la pente de surface.

Le critère principal utilisé pour détecter les obstacles est la pente d'une surface. Un obstacle positif n'est qu'une surface dont la pente est inclinée.

Pour un terrain plat,la différence dans les coordonnées sur l'axe x entre successives pixels n'est qu'une fonctionne croissante. À cause de la résolution décroissante de l'image, le terrain visible dans l'image de *LADAR* peut être divisé en deux régions : basse résolution et haute résolution. La région de haute résolution de l'image contient des points terrestre finement échantillonnés alors que la région de basse résolution représente un échantillonnage grossier du terrain. La détection des obstacles négatifs représente une tâche difficile par rapport aux obstacles positifs. Cela vient du fait que les obstacles négatifs n'ont pas de pente inclinée en raison de l'occlusion. Les trous sont considérés comme une grande discontinuité négative dans l'élévation. Pour réduire les fausses alarmes, les trous ne sont pris en considération que dans la région de haute résolution.

Suzuki *et al*. (2010) a utilisé, à la place d'un capteur GPS, une caméra et un laser pour concevoir une méthode de navigation à l'extérieur. L'étude a été divisée en deux sous-problèmes :

1. la réalisation d'une cartographie simultanée soutenue par une estimation de la position de robot comme dans la méthode de SLAM (Simultaneous Localization and Mapping) ;
2. l'estimation de la position en faisant la correspondance entre la carte courante et les données du senseur.

Pour réaliser ce travail, une carte $3D$ qui utilise un point $3D$ acquis par le MMS (Mobile Mapping System) et propose une estimation de $6ddl$ (degrés-de-liberté) de la position et l'attitude en référence aux pneus du robot. La méthode pour estimer la position est composée d'une cartographie $3D$ avec MMS et localisation de robot en utilisant ces cartes préparées (construites en temps réel).

Pour détecter les objets mobiles, Jung et Sukhatme (2004) préfèrent utiliser une caméra pour la navigation à l'extérieur, mais cela n'est pas une tâche facile, car il y deux mouvements, l'obstacle et le robot. Le mouvement de caméra est compensé en utilisant les relations entre le repère de la caméra et le repère du robot (l'orientation de la caméra est connue par rapport au robot). L'estimation de la position des objets qui bougent est estimée en utilisant un filtre à particule adaptatif et itératif et un algorithme EM (Expectation-Maximization). Le principe sous-jacent est de faire la différence entre deux images consécutives. L'algorithme est rapide et efficace dans le cas où le domaine de vue de caméra est statique. Quand la caméra bouge, la différence directe n'est plus applicable en raison d'une grande différence générée par un mouvement simple de la caméra même si rien ne bouge dans l'environnement. Étant donné qu'on a deux types de mouvements qui sont intégrés dans une seule image, il faut éliminer le mouvement de caméra afin de détecter le mouvement des objets qui bougent dans la scène. Comme tous les problèmes de navigation à l'extérieur, il y a toujours des sources de bruits beaucoup plus importantes que dans la navigation à l'intérieur. Alors l'image est contaminée par plusieurs sources de bruit comme les conditions de climat, le bruit de caméra et

un changement dans les forme des obstacles. Un bruit est attaché spécialement aux frontières des objets à cause du manque des informations de la profondeur provenant d'un caméra monoculaire. Certains types de bruit sont transitoires et d'autres sont invariants dans le temps. Un modèle probabiliste est indispensable pour filtrer ce bruit et pour faire une robuste détection et un bon suivi des objets.

Lee et Song (2004) ont porté leur attention sur les erreurs des paramètres du module cinématique : le glissement et une surface inégale peuvent engendrer des erreurs dans l'estimation de la position. La piètre estimation de la position durant l'exécution du trajet exige une localisation fréquente à l'aide de références externes. Une mauvaise estimation peut amener à une collision dans le cas de la présence d'un obstacle. Deux méthodes, se basant sur l'utilisation des senseurs optiques pour bien estimer les erreurs de la position, sont suggérées dans cette études. Les erreurs de l'odométrie peuvent être calculées en faisant des localisations fréquentes mais cela doit être fait par des senseurs supplémentaires.

L'étude commence avec un seul senseur optique en prenant les contraints cinématiques en considération pour prouver le principe ensuit deux senseurs optique de même type ont été utilisés pour avoir un système de localisation plus robuste aux perturbations en négligeant les contraints cinématiques.

La méthode avec un seul senseur optique fournit la position en x et y mais pour calculer la vitesse de rotation, on doit calculer $\frac{\Delta x}{\Delta t}$ et $\frac{\Delta y}{\Delta t}$. Il est supposé que le centre du robot bouge dans une direction perpendiculaire à l'axe de la roue. Par la suite, il faut calculer la vitesse des roues et les transférer dans le repère du robot. Les coordonnées x,y et θ dans le repère de robot sont calculées en intégrant les vitesses linéaires et angulaires du robot.

L'expérimentation pratique montre de bons résultats en éliminant les contraintes cinématiques et en utilisant deux senseurs optiques.

1.3 Méthodes basées sur l'utilisation d' un GPS

Panzieri *et al*. (2002) ont fait des études sur la précision d'un capteur GPS commercial. Une carte a été utilisée pour aider le robot à se localiser lorsque le capteur a de grandes erreurs. Ils ont constaté que le nombre de satellites n'a pas un effet important sur la précision de capteur, pourtant le nombre minimum des satellites pour avoir des données GPS pertinentes est 3 (pour un véhicule au sol). Un autre facteur important pour ce processus est le DOP (dilution of precision), qui détermine la géométrie des satellites détectés par le capteur, ce dernier indique si le nombre de satellites est une indication importante de la précision ou bien non. Le DOP n'est pas utile lorsque les satellites sont alignés, dans ce cas-ci le nombre de satellites ne reflète pas la réalité. Dans ce travail, ils ont essayé de relier le nombre de satellite avec l'écart type de donnés requis mais en prenant le DOP en

considération. Par contre, ils ont trouvé que le changement de l'écart type de GPS n'a pas de grand effet sur le processus de positionnement.

Les résultats attendus n'étaient pas assez bons pour réaliser un positionnement acceptable. C'est pour cela qu'ils ont utilisé quelques repères dans cet environnement pour suivre un trajet désiré. Les outils mathématiques utilisés dans ce travail sont le filtre de Kalman étendu pour fusionner les données de GPS et l'odométrie.

Kim *et al*. (2007) ont proposé un modèle pour la navigation à l'extérieur qui est composé de deux modèles ; un modèle autonome qui est responsable de la navigation dans le cas où il n'a pas d'obstacles et un modèle guidé par un usager est utilisé dans les autres cas. Pour réaliser leur travail, ils ont fusionné le DGPS (Differential Global Positioning System) avec l'odométrie par le filtre de kalman étendu. La détection des dénivellations et des obstacles a été faites par un laser incliné. La détection des dénivellations dans une rue est réalisée par la transformation de *Hough*, utilisée afin de calculer le rayon et angle dominant d'une ligne droite, pour déterminer la fin d'une ligne droite (la ligne de contact entre les rayons laser et la surface de la rue). Cette fin représente le début d'une dénivellation.

Dans leur travail, Ohno *et al*. (2004) ont utilisé le DGPS à la place d'un GPS (RTK-GPS) (Real-Time Kinematic-GPS), bien que ce dernier soit assez précis, pour deux raisons essentielles ; la première est l'insuffisance du nombre de satellites requis pour donner une position exacte du robot en raison des arbres et des bâtiments. Dans ces conditions, le GPS (RTK-GPS) n'est plus capable de donner une position à cause de la discontinuité dans le signal GPS. La deuxième raison est le multi-trajet causé par la réflexion du signal GPS. Des critères existent pour déterminer l'exactitude comme le DOP (dilution of precision) et le nombre des satellites ; cependant en pratique, cela ne reflète pas la réalité. La première cause a été surmontée en utilisant le DGPS ; cependant la qualité des mesures DGPS (Differential Global Positioning System) a été évaluée par la méthode proposée qui utilise l'odométrie. Il est supposé que l'erreur de l'odométrie est petite pour de courtes distances. Autrement-dit, il est supposé que le trajet de l'odométrie est continu sur le trottoir et qu'il n'y a pas de glissement. Pour déterminer l'exactitude de données DGPS, les log-likelihood de la position, de l'orientation de DGPS (l_xy, l_θ) sont calculés par l'équation de distance de Mahalanobis. Par la suite, si le log-likelihood (lxy, l_θ) est plus petit qu'un seuil, alors les données DGPS sont considérées comme précises.

1.4 Conclusion

Dans la revue de littérature précédente, des méthodes intéressantes sont utilisées pour réaliser une navigation autonome ou semi-autonome. Certains ont utilisé des caméras pour distinguer les obstacles et pour localiser le robot dans une carte 2D ; un système de vision peut être utilisé pour

construire une carte et pour aider le robot à se localiser dans l'environnement où il fait son parcours. Ce type d'appareils est beaucoup plus efficace à l'intérieur qu'à l'extérieur en raison du bruit. La navigation à l'extérieur complique la tâche à faire à cause des phénomènes qui peuvent survenir comme des obstacles mobiles, un changement du climat, des travaux, un trottoir déformé etc. Les appareils candidats pour détecter les obstacles sont le sonar, le télémètre laser, le radar, etc. Par contre, pour obtenir une architecture commerciale de système robotique, le radar n'est pas un bon choix (par son coût) ; le télémètre laser peut-être considérée comme un choix acceptable. Dans notre projet de recherche, on a utilisé le laser pour détecter les dénivellations de trottoir et pour garder le robot à une distance sécuritaire des dénivellations et des obstacles fixes et mobiles. Les recherches faites pour la navigation urbaine utilisent des appareils GPS assez précis ; certains utilisent DGPS pour localiser le robot dans les quartiers résidentiels. Un GPS commercial a été impliqué dans notre projet pour localiser le robot à l'extérieur. Dans les chapitres suivants, nous allons suggérer des algorithmes pour projeter les données GPS et corriger les erreurs associées et d'autres algorithmes pour éloigner le robot des obstacles et du danger.

CHAPITRE 2

LOCALISATION GLOBALE ET LOCALE PAR GPS

2.1 Introduction

Dans le cadre d'une navigation semi-autonome d'un robot mobile dans un environnement urbain, l'utilisation du GPS pour la localisation pourra apporter des informations importantes et complémentaires aux autres capteurs pouvant être utilisés pour la planification et le suivi de chemin. Dans ce chapitre, nous allons présenter différentes expériences réalisées afin de caractériser un capteur GPS afin de connaître le niveau de précision obtenu et les caractéristiques de erreurs.

La localisation de notre système s'est faite à l'aide d'un senseur GPS qui aura, entre autre, pour rôle d'assister l'usager dans la navigation en milieu urbain (sur une route piétonnière). En principe, le senseur GPS utilisé devrait être assez précis pour localiser le robot. Comme nous le verrons plus loin, le capteur « commercial » utilisé entraînait une marge d'erreur relativement élevée (plus grand que la largeur d'un trottoir ou d'une rue par exemple). Pour palier à cette difficulté, il a été suggéré une méthode visant à diminuer les erreurs (en constatant que celle-ci était constituée d'un biais variant lentement et d'un composante aléatoire) et à améliorer la précision du GPS. Cette dernière est bien détaillée dans le cadre de la localisation des données GPS dans un repère local. Les résultats obtenus au cours de l'expérimentation ont été satisfaisants et ont vérifié la pertinence de cette analyse des erreurs dues aux imprécisions inhérentes au senseur GPS. La localisation globale quant à elle est faite par un autre appareil GPS qui donne le parcours référence à emprunter. Une solution pour amener le robot vers ce chemin désiré a été développée dans les sections ci-dessous.

2.2 Problématique de la localisation du robot

Dans la littérature, plusieurs méthodes suggèrent l'utilisation de caméras avec un laser pour localiser le robot dans sont environnement (Wijesoma *et al.*, 2002; Truong *et al.*, 2007). Cependant ces méthodes restent insatisfaisantes pour un environnement extérieur urbain de par sa grande dimension, et de la difficulté d'obtenir un système à coût raisonnable (complexité et lourdeur du traitement des données). Le fait de réduire le coût d'un système de navigation le rend plus accessible, et son utilisation pratique envisageable à grande échelle. L'introduction d'un senseur GPS de haute qualité dans des applications de navigation induit certes une bonne précision, mais d'un autre côté génère un coût additionnel. C'est pour cela que le senseur GPS utilisé dans cette application est un appareil commercial bon marché, entraînant une incertitude de mesure non négligeable au

demeurant. De ce fait, la localisation devient plus difficile d'autant plus que la fusion de données nécessite un appareil assez précis pour réduire l'erreur de la dynamique du robot. La localisation par ce senseur est soumise alors à certains types d'erreurs qui sont répertoriés et traités dans un bon nombre de publications orientées vers l'amélioration de la précision (Panzieri *et al*., 2002; Abdullah *et al*., 2010; Maier et Kleiner, 2010). Par conséquent pour compenser l'absence d'un système de navigation de haute qualité, l'alternative logicielle consistant à développer un algorithme stabilisant puis corrigeant les erreurs de mesure est disponible, solution présentée dans le présent chapitre.

Le parcours référence est extrait d'un appareil GPS Nuvi de Garmin qui permet d'obtenir un chemin piétonnier (la trajectoire) dans un format standard : GPX. Il est important de bien définir cette route puisqu'elle permet de construire un chemin virtuel à parcourir (notre chemin réel sera relativement parallèle à celui-ci). À la fin du chapitre, il sera proposé des solutions pour résoudre ce problème.

2.3 Description sommaire du système GPS

Le système GPS (Global Positioning System) conçu en premier par le département de la défense américaine pour améliorer les systèmes de navigation militaires (aériens et maritimes) existants à l'époque et limités par des contraintes de type météorologiques, astrales et stratégique, a été homologué comme un moyen de radionavigation utilisable par les civils en 1993.

Le premier satellite d'un prototype GPS date de 1972(Wikipedia, 2010b), année à laquelle le premier satellite comportant une horloge atomique est mis en orbite. En 1998 (Wikipedia, 2010b), les aéronefs de transport civils sont autorisés à utiliser le système GPS. Il est alors décidé de le scinder en deux services : Le PPS (Precise Positioning System), service de localisation très précis et réservé au corps militaire, le SPS (Standard Positioning System), service de localisation aux performances dégradées utilisable par les civils. Hormis les avantages évoqués plus haut pour lesquels le GPS a été conçu, il peut aussi lui être reconnu :

- une simplicité d'emploi,
- une précision accrue comparée à celle des systèmes de navigation classique,
- un faible coût du récepteur.

Le système GPS est composé sommairement de trois parties distinctes :

1. des satellites en orbite autour de la terre,
2. des stations au sol qui contrôlent ces satellites et traitent les informations transmises par ces derniers,
3. des récepteurs à la disposition des utilisateurs.

2.4 Principe de fonctionnement de GPS

Cette section présente un bref résumé des informations disponibles dans les références (Pièplu, 2006; Du Puy de Goyne, 2003).

2.4.1 Mesure de la distance satellite récepteur

Le récepteur GPS mesure sa distance par rapport à un satellite dont la position est bien connue par ce récepteur. Cette distance est obtenue par la mesure du temps mis par une onde radio pour se propager du satellite jusqu'à l'antenne du récepteur.

$$\text{Distance} \quad = \quad \text{temps} \cdot \text{vitesse}$$

Cette distance est nommée (pseudo-distance) car elle est mesurée par la différence de temps entre deux horloges différentes : une dans l'émetteur, l'autre dans le récepteur. Une première problématique consiste à synchroniser les horloges puisque la précision de la mesure de distance en dépend directement.

Les données émises par le satellite informent le récepteur de :

- la position du récepteur dans l'espace,
- l'horloge du satellite,
- et l'état de fonctionnement du satellite.

Pour fixer un ordre de grandeur, supposons que la distance entre les deux récepteurs soit de 8000 kilomètres.

$$\alpha \quad = \quad \frac{\beta}{\gamma} = \frac{8000}{300000} = 0.026 \text{ sec}$$

où

α : l'écart de temps

β : l'écart de distance

γ : la vitesse

Ce calcul est plus intéressant sur des écarts de distance fiables afin de mettre en évidence la précision et la sensibilité de chronométrage imposée au récepteur. Pour détecter un écart d'un kilomètre, ce qui est un écart important par rapport à la précision du système GPS, le récepteur doit pouvoir mesurer un écart type de temps égal à :

$$\frac{1}{300000} \quad = \quad 3.33 * 10^{-6} \text{ sec}$$

Soit environ 3 millionièmes de secondes.

2.4.2 Position calculée dans le système WG84

À partir des ondes radioélectriques reçues des satellites, le récepteur GPS calcule une position en coordonnées géographiques par rapport au système géodésique de référence mondiale et nommé WG84 (World Geodetic System 1984). Ce système calcule les orbites des satellites à partir des mesures des stations de contrôle.

2.4.3 Définitions

Latitude et parallèles :

La latitude (Du Puy de Goyne, 2003) est une coordonnée géographique, représentée par une valeur angulaire, exprimant la position Nord-Sud d'un point sur terre par rapport à l'équateur. Elle est, en navigation, l'angle que fait la normale à l'ellipsoïde de référence avec le plan équatorial. Chaque point à la surface du globe terrestre peut être situé sur un cercle, nommé Parallèle et constitué de tous les points ayant la même latitude.

Longitude et méridiens :

La longitude(Du Puy de Goyne, 2003) est une coordonnée géographique, représentée par une valeur angulaire, exprimant la position Est-Ouest d'un point sur terre par rapport au méridien de origine soit le méridien de Greenwich passant par l'observatoire de Londres.

Note : l'équateur est l'origine des parallèles et latitudes. Il est fixé par la forme géométrique de la Terre (plus grand parallèle possible perpendiculaire à l'axe de rotation de la Terre ou axe des pôles), Alors que le méridien de Greenwich, origine des méridiens et des longitudes, a été fixé par convention.

Relèvement :

Le relèvement (Wikipedia, 2010a) est l'angle entre le Nord vrai ou géographique et un point donné par rapport à un point intermédiaire.

Relèvement vrai :

Le relèvement vrai d'un point $B(lat2, lon2)$ par rapport à un point $A(lat1, lon1)$ est l'angle entre la direction du Nord géographique et le segment $[BA]$.

Relèvement magnétique :

Le relèvement magnétique d'un point B par un point A est l'angle$(\vec{AN}, \vec{AB})$ où N est le nord magnétique. Par rapport au relèvement vrai, il intègre la déclinaison magnétique (Wikipedia, 2010a).

La formule est :

$$\theta = \text{atan2}(\sin(\Delta long)\cdot\cos(lat2), \cos(lat1)\cdot\sin(lat2) - \sin(lat1)\cdot\cos(lat2)\cdot\cos(\Delta long))$$

où

$$\Delta long = Long2 - Long1$$

Étant donné que atan2 retourne des valeurs entre

$$[\pi, -\pi]$$

pour normaliser le relèvement entre 0 et 360° ;il faut convertir en degré et appliquer la formule :

$$(\theta + 360)\%360$$

où % c'est le modulo. Le θ est le prélèvement initial qui est suivi au cas d'une ligne droite à travers l'arc de grand cercle. Si on se déplace en suivant ce relèvement, cela va nous amener du point de départ au point de destination. La figure 2.1 montre le relèvement de deux points A,B.

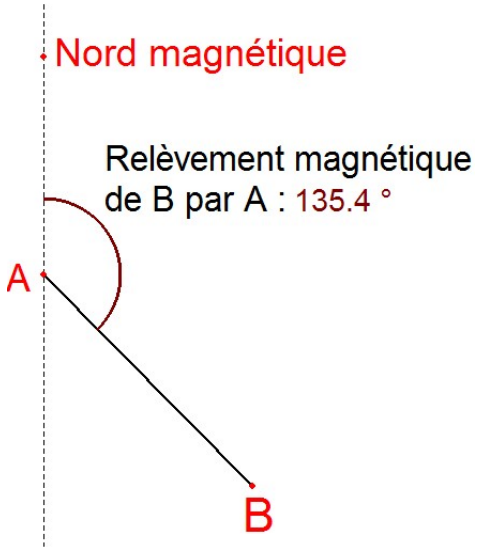


Figure 2.1 Relèvement magnétique

2.5 Format GPX

Le format GPX ou GPS exchange est un schéma XML désigné pour présenter les données GPS d'une façon propice aux logiciels d'applications de GPS et aux logiciels de navigations (Foster, 2002). Ce format peut être utilisé pour décrire les points d'intérêts, trajets et routes. Il est ouvert et peut être utilisé sans besoin d'une licence d'utilisation. Dans ce type de fichier, on retrouve les coordonnées, élévation, et le temps correspondant à chaque échantillon. Il peut être utilisé pour échanger les données entre les appareils GPS et différents logiciels. Ce type de fichier nous permet de visualiser les parcours suivis et enregistrés par l'appareil. La collecte des points, dans le format GPX, organisée séquentiellement, constitue soit des routes ou bien des trajets. Conceptuellement, les trajets sont des way points (des points entrés en mémoire constituant une trajectoire) où la personne a navigué. Les routes sont des suggestions quant aux chemins éventuels à emprunter. Ces dernières sont sujettes à être modifiées lors de la navigation en temps réel.

La différence entre ces deux concepts à été établie en vue de justifier la présence de commandes spéciales dans tel type d'approche de la navigation. À titre d'exemple, citons le *time stamp* valable pour chaque point de trajet, qui reste cependant valable que pour des points particuliers de la route. Les propriétés de base caractérisant un fichier GPX sont la latitude et la longitude associées à chaque point. Les autres variables sont optionnelles.

2.6 Schéma GPX

Un fichier GPX (company Garmin, 2010) contient les éléments suivants :

1. Espace de noms de cible (*Target Namespace*).
2. Espace de noms des éléments et attributs (*Element and Attribut Namespaces*).

L'entête de chaque fichier GPX contient un espace de noms (*Namespace*), la version du dispositif et la définition du type GPX utilisé par la compagnie fournissant le matériel en l'occurrence Garmin.

Pour une route : La racine principal de l'arbre commence par `<rte>` et se finis par `</rte>` Entre les deux suffixes `<rte>`, il y a le suffixe `<name>` qui contient la destination. Le suffixe `<rtep>` contient les coordonnées latitude et longitude des points de départ et d'arrivée (voir figure 2.2).

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<gpx xmlns="http://www.topografix.com/GPX/1/1"
    xmlns:gpxx="http://www.garmin.com/xmlschemas/GpxExtensions/v3"
    xmlns:gpxtpx="http://www.garmin.com/xmlschemas/TrackPointExtension/v1"
    creator="nÃ¼vi 750" version="1.1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.topografix.com/GPX/1/1
    http://www.topografix.com/GPX/1/1/gpx.xsd
    http://www.garmin.com/xmlschemas/GpxExtensions/v3
    http://www.garmin.com/xmlschemas/GpxExtensions/v3/GpxExtensionsv3.xsd">
  <metadata>
    <link href="http://www.garmin.com">
      <text>Garmin International</text>
    </link>
    <time>2010-11-08T01:35:33Z</time>
  </metadata>
  <rte>
    <name>5855 Rue Souart from Avenue Ellen &amp;amp; </name>
    <rtept lat="45.502883" lon="-73.626338">
      <name>Avenue Ellendale &amp;amp; R</name>
      <extensions>
        <gpxx:RoutePointExtension>
          <gpxx:Subclass>000093dd77100220e8080f20cb008c5bbca4</gpxx:Subclass>
          <gpxx:rpt lat="45.502883" lon="-73.626338">
            <gpxx:Subclass>000093dd770001d5100021160000b6000000</gpxx:Subclass>
          </gpxx:rpt>
          <gpxx:rpt lat="45.502883" lon="-73.626338">
            <gpxx:Subclass>000093dd770001d510001f0009008ae62500</gpxx:Subclass>
          </gpxx:rpt>
          <gpxx:rpt lat="45.502196" lon="-73.626982">
            <gpxx:Subclass>000093dd77006e3e05001f02090040e71500</gpxx:Subclass>
          </gpxx:rpt>
          <gpxx:rpt lat="45.502453" lon="-73.627497">
            <gpxx:Subclass>000093dd77006e3e0500211700001500000</gpxx:Subclass>
          </gpxx:rpt>
        </gpxx:RoutePointExtension>
      </extensions>
    </rtept>
    <rtept lat="45.502475" lon="-73.627475">
      <name>5855 Rue Souart</name>
      <extensions>
        <gpxx:RoutePointExtension>
          <gpxx:Subclass>000093dd77006e3e050001a000005a140000</gpxx:Subclass>
        </gpxx:RoutePointExtension>
      </extensions>
    </rtept>
  </rte>
</gpx>
```

Figure 2.2 Fichier GPX

En général, les applications permettent la conversion vers le format GPX ou à partir du format GPX vers un autre format. Certaines permettent aussi de télécharger ou de transmettre des données GPX à un autre appareil GPS. Pour visualiser les routes et trajets, GPS Visualiser est utilisé à l'aide des cartes fournies par Google. Ce site permet de visualiser et changer un fichier d'un format supporté pour le convertir en un autre format : GPX, texte, JPEG, KML, ...

2.7 Senseur GPS 18 5Hz

Le Senseur GPS 18 5Hz (Garmin, 2010) est utilisé avec des systèmes de contrôle, guidage et les applications d'agriculture qui demandent un compte rendu contenant position et vitesse à chaque 5Hz à partir d'un récepteur GPS. Ce senseur possède 12 canaux parallèles avec un WAAS activé (*Wide Area Augmentation System — WAAS-enabled*). Le GPS 18 5Hz (figure 2.3) mémorise les configurations dans une mémoire non volatile. Il a aussi une horloge à temps réel et une sortie de mesures en série pour des applications et offre une sortie de fréquence 5Hz avec des fronts montants qui qui sont coordonnés avec l'UTC. Le senseur émet les données en forme de phrases appartenant au protocole NMEA et à un protocole de **Garmin**.

Figure 2.3 Le Senseur GPS 18 5Hz

2.8 Protocole NMEA

Ce protocole est une norme pour la communication entre équipements maritimes dont les équipements GPS. Elle est définie et contrôlée par la National Marine Electronics Association (NMEA). La norme 0183 utilise une simple communication en série pour transmettre une "phrase" à un ou plusieurs écoutants. Une trame NMEA utilise tous les caractères ASCII. Les phrases utilisées dans notre senseur sont :

AAM - Waypoint Arrival Alarm.

ALM - Almanac data.

APA - Auto Pilot A sentence.

APB - Auto Pilot B sentence.

BOD - Bearing Origin to Destination.

BWC - Bearing using Great Circle route.

DTM - Datum being used.

GGA - Fix information.

GLL - Lat/Lon data.

GRS - GPS Range Residuals.

GSA - Overall Satellite data.

GST - GPS Pseudorange Noise Statistics.

GSV - Detailed Satellite data.

MSK - send control for a beacon receiver.

MSS - Beacon receiver status information.

RMA - recommended Loran data.

RMB - recommended navigation data for gps.

RMC - recommended minimum data for gps.

RTE - route message.

TRF - Transit Fix Data.

STN - Multiple Data ID.

VBW - dual Ground / Water Speed.

VTG - Vector track an Speed over the Ground.

WCV - Waypoint closure velocity (Velocity Made Good).

WPL - Waypoint Location information.

XTC - cross track error.

XTE - measured cross track error.

ZTG - Zulu (UTC) time and time to go (to destination).

ZDA - Date and Time.

Les phrases utilisées dans notre application (un *plugin* `GpsData` développé pour intégration dans Acropolis, voir annexe B pour une description de l'architecture de commande) sont GPGGA, PGRME, GPVTG.

Le *plugin* est conçu pour extraire les données de position (latitude, longitude),vitesse et les erreurs (erreur de longitude, erreur de latitude) ainsi que plusieurs attributs caractéristiques comme PVT qui donne la vitesse en kilomètres. En voici un exemple.

`$-GGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx`

- hhmmss.ss = UTC de la position
- llll.ll = latitude
- a = N or S
- yyyyy.yy = longitude
- a = E or W
- x = indice de qualité de Gps (0=no fix, 1=GPS fix, 2=Dif. GPS fix)
- xx = nombre de satellite en usage
- x.x = dilution horizontale de précision
- x.x = altitude au dessus de la mer
- M = unités de l'altitude de l'antenne, en mètre
- x.x = séparation de Géoïde
- M = unité de séparation, en mètre
- x.x = âge de données Gps Différentiel (en secondes)
- xxxx = référence différentielle

La compagnie Garmin a créé son propre protocole et les phrases dans ce protocole sont GPRMC, GPGGA, GPGSA, GPGSV, PGRME, GPGLL, GPVTG, PGRMV, PGRMF, PGRMB, PGRMT.

`$PGRME,<1>,M,<2>,M,<3>,M*hh<CR><LF>`

<1> l'estimé de l'erreur horizontale de la position (HPE), 0.0 to 999.9 mètres.

<2> l'estimé de l'erreur verticale de la position (VPE), 0.0 to 999.9 mètres.

<3> l'estimé de l'erreur de la position (EPE), 0.0 to 999.9 mètres.

Ce ne sont pas toutes les phrases qui seront utilisées dans notre *plugin* mais seulement celles qui se sont avérées utiles pour notre application. Lors de l'extraction des données, il sera important

de prendre en considération la synchronisation entre le *plugin* `GpsData` d'Acropolis de fréquence 10Hz et la fréquence du senseur de 5Hz.

2.9 Expérimentation de senseur GPS

Dans une première série de tests, nous allons parcourir des trajectoires connues. La position obtenue en temps réel pourrait avoir une précision allant de 3m, dans le cas de la présence d'un signal *WAAS*, et jusqu'à 10m, en absence du signal de correction. Ces tests ont été effectués dans un quartier résidentiel (pas d'édifices en hauteur : rue Souart et avenue Ellendale). Les trajectoires correspondent à des déplacement des deux côtés de la rue : trottoirs de droite et de gauche. Ceci afin de :

- savoir où sont placées les points du fichier GPX généré en mode piétonnier,
- analyser les résultats en terme d'erreurs,
- suggérer une méthode pour minimiser les erreurs de GPS.

Divers tests auront été effectuées sur une vingtaine de jours. Dans un premier cas, un point d'origine et une destination sont utilisés pour générer une route. Dans un premier test,la première partie de la trajectoire s'est révélée être sensiblement conforme à la course, de plus elle était plus rectiligne que la deuxième partie du trajet. Le point de départ est proche du point réel même si le point d'arrivée reste éloigné de 1 mètre de sa position réelle. La figure 2.4 montre les résultats d'un test sur le trottoir droite de la rue. On voit clairement que l'évolution des longitudes et latitudes permet de positionner la trajectoire sur la « bonne route », mais pas nécessairement du bon côté.

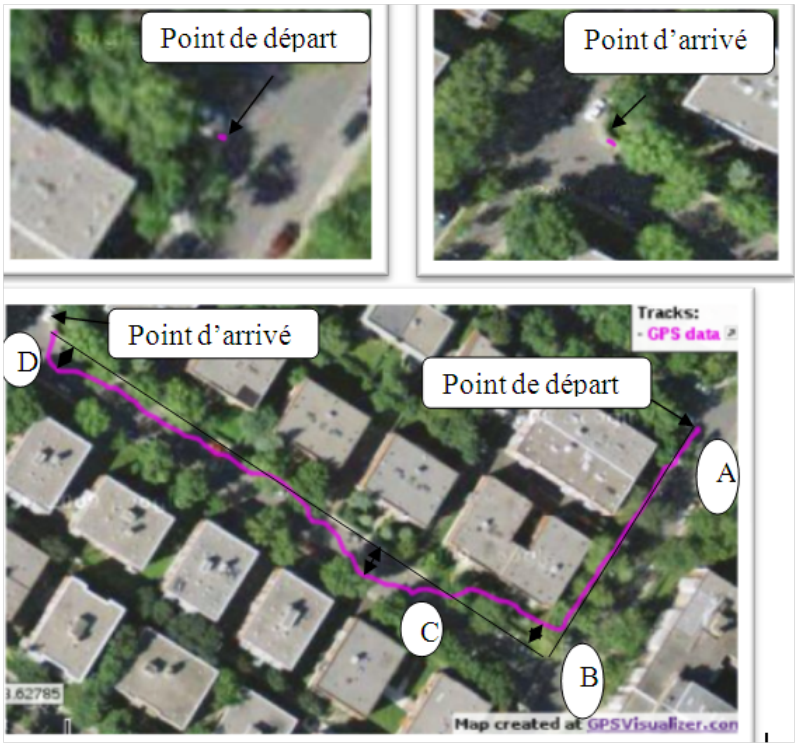


Figure 2.4 Test sur le trottoir de droite d'une rue

Un deuxième trajet du côté gauche du trottoir est présenté à la figure 2.5. Les résultats montrent

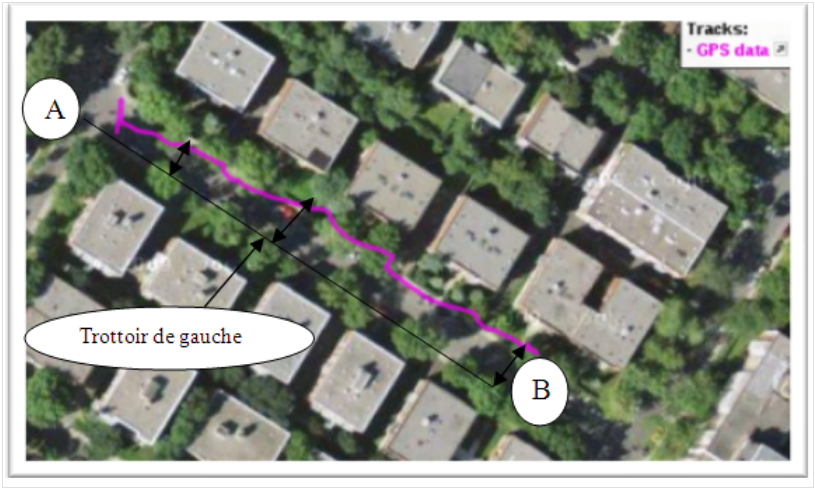


Figure 2.5 Test sur le trottoir de gauche d'une rue

des erreurs plus grandes en longitude, et donc l'erreur de déplacement sur le trottoir du gauche est plus grande que de celui de droite. Une cause probable de ces plus grandes erreurs est que des arbres cachent une partie importante du trottoir de gauche dans le deuxième test ce qui dégrade la

réception du signal.

La figure 2.6 montre bien que les points d'une route de la carte numérique sont pris au milieu de la rue et en théorie ces points devraient être assez précis. Les trois courses sont présentées dans la même figure (2.7). Il est clair que seul le GPS non corrigé ne pourra assurer une navigation à l'extérieur.

Figure 2.6 Planification d'une route à partir d'un fichier GPX

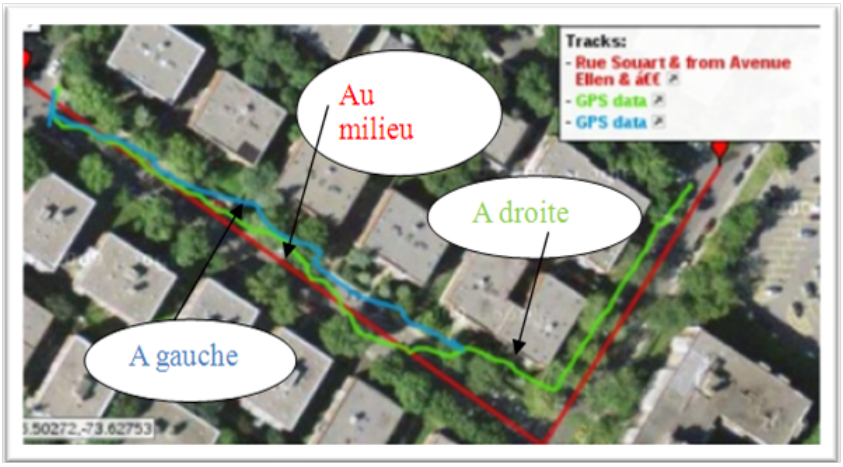


Figure 2.7 Comparaison des trajectoires suivies à gauche et à droite

2.10 Caractérisation des erreurs

Dans la section précédente, nous avons pu constater que les erreurs du GPS pouvaient nous induire en erreur quand au côté de la rue sur lequel le système se déplaçait. L'objectif de la prochaine serie de tests est de caractériser les erreurs faites par le senseur GPS. Selon les spécifications, les

erreurs anticipées sont de 15m s'il n y pas de système WAAS et de moins de 3m quand le WAAS est disponible. Mais après avoir fait recueilli des données pendant vingt jours (matin et soir), nous avons constaté que l'erreur est plus grande que celle fournie dans les spécifications.

D'abord, nous avons placé le GPS en un point fixe ($lat = 45.50231°, lon = -73.62751°$) pour de très longues périodes. Les paramètres observés sont la longitude et la latitude. Pour chacune des mesures, on calcule la distance entre le point du test et le point mesuré ainsi que la différence entre la longitude et la latitude.

Les mesures ont été enregistrées à partir de 9h du matin, on remarque que l'erreur est presque aléatoire mais que cette dernière est compensable en matinée alors qu'elle ne l'est pas en après midi. De plus, l'erreur de différence de la longitude et de la latitude pourrait nous donner une idée des erreurs intégrée dans le signal de GPS.

Les points consécutifs pris par le senseur pour un seul point sont bien montrés à la figure 2.8. Le senseur GPS senseur nous donne l'impression que le point de test se déplace environ 5 cm à chaque lecture, alors quand le robot est immobile.

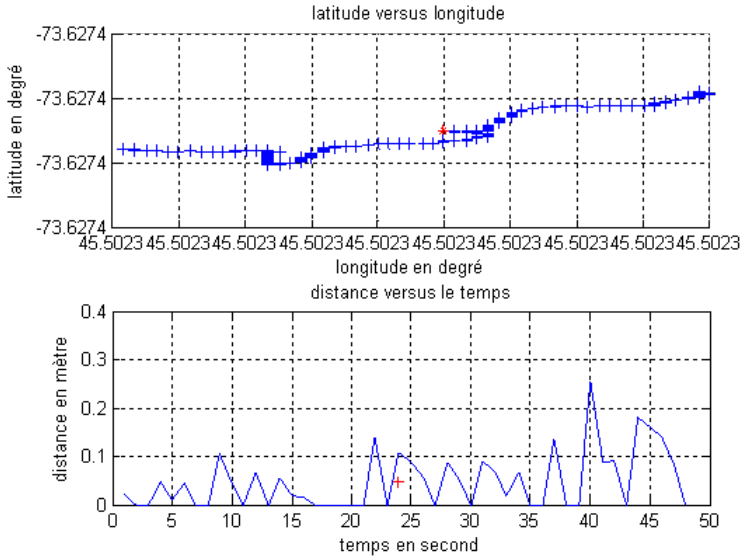


Figure 2.8 Position mesurée et incrément de distance entre les points donnés par le senseur (point fixe)

La figure 2.10 montre l'erreur de la latitude, on remarque que l'erreur est aléatoire mais corrélée dans le temps.

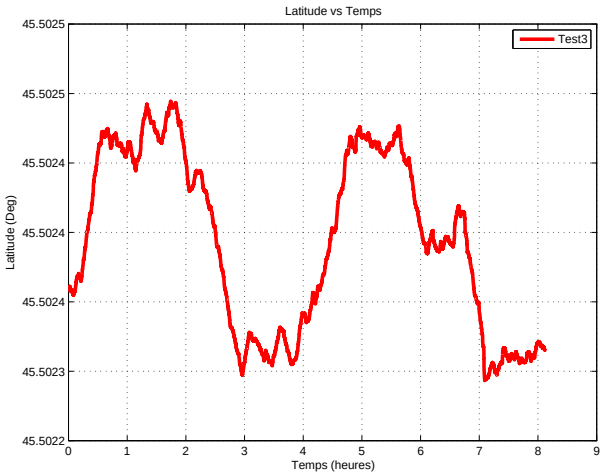


Figure 2.9 Latitude

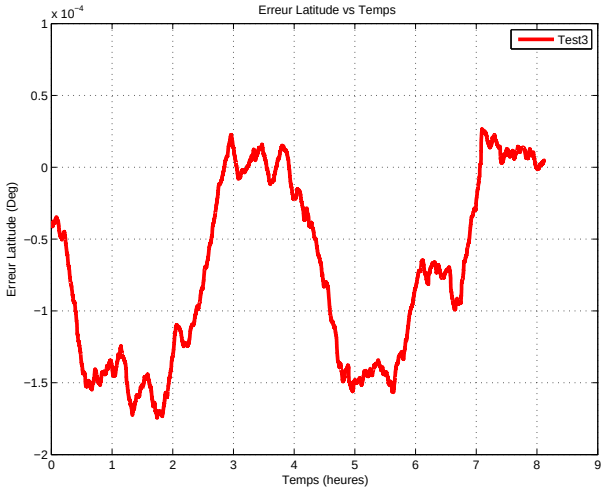


Figure 2.10 Erreur correspondante de la latitude

Après avoir répété le test dans une période allant du matin jusqu'au soir pour plusieurs jours, on peut remarquer que les erreurs se répétaient d'une manière similaire.

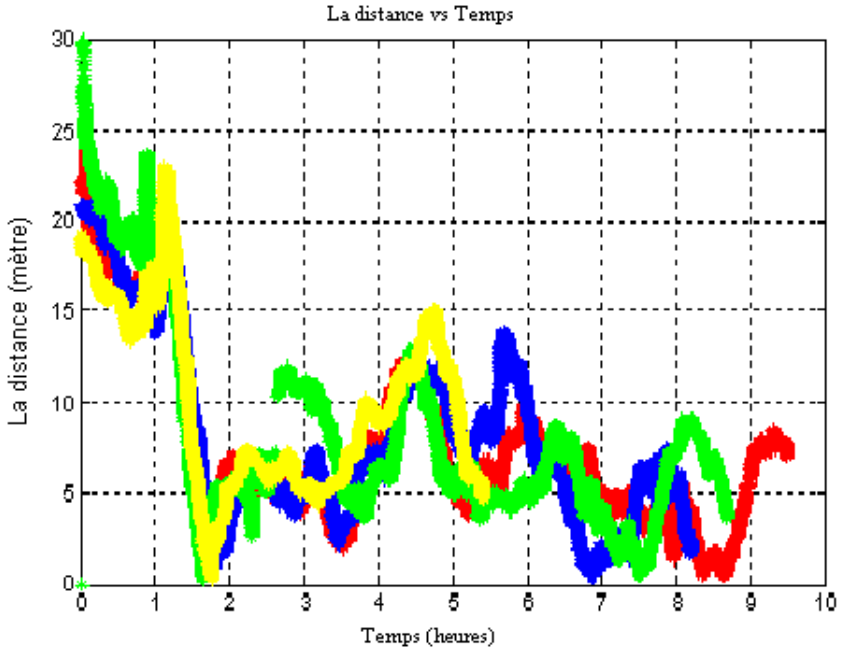


Figure 2.11 Erreur de la distance pendant une journée de test

La figure 2.11 montre bel et bien le bruit impliqué dans le signal de senseurs GPS pendant plusieurs jours consécutifs ensoleillés.

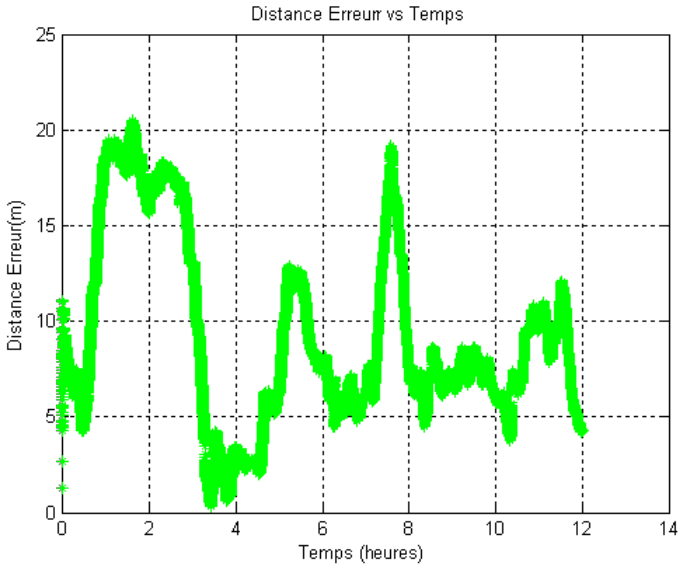


Figure 2.12 Erreur de la distance pour une période de 12 heures

En utilisant un de ce groupe des tests, la figure 2.12 représente une période de 7h30h à 19h30. On remarque que la partie entre 4 et 6 se répète chaque jour et l'erreur se rapproche de zéro. Donc on va considérer que l'erreur de GPS ne varie trop pendant les tests.

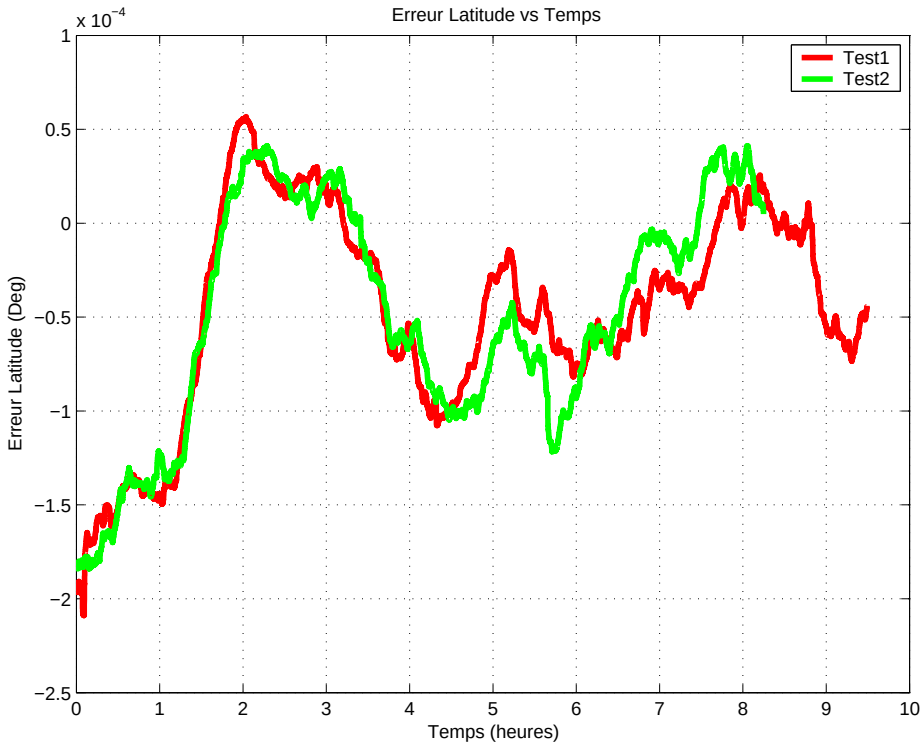


Figure 2.13 Erreur de la latitude pour deux tests

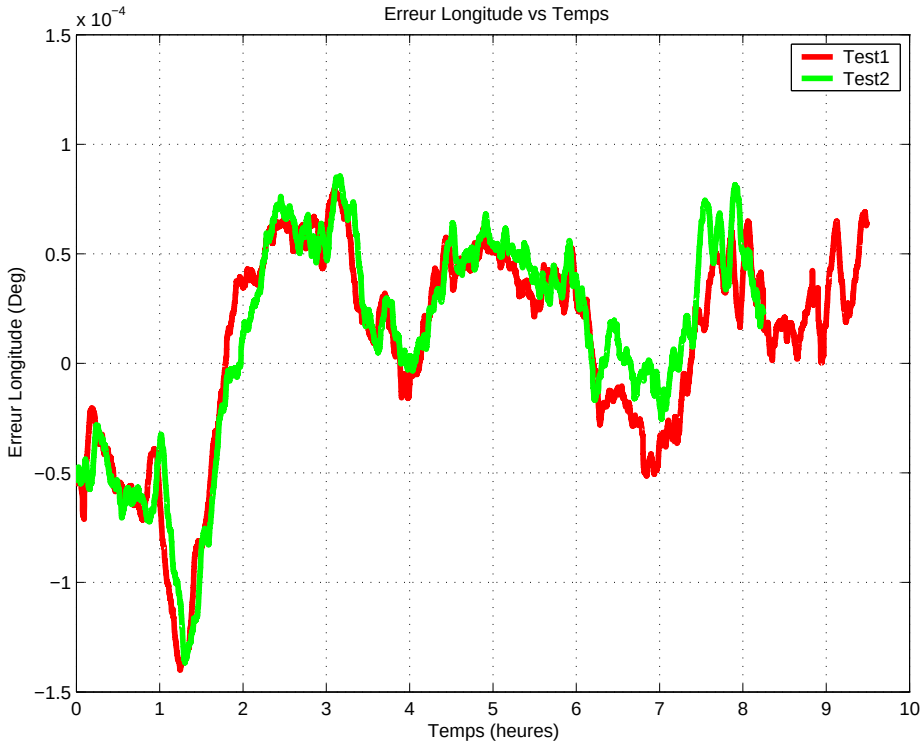


Figure 2.14 Erreur de la longitude pour deux tests

La figure 2.13 montre une comparaison de l'erreur sur la latitude. Les tests sont faits pendant deux jours différents durant la même période de la journée, on remarque qu'il y une conformité entre les erreurs de la longitude et la latitude. Les figures suivantes sont des comparaisons additionnelles pour l'erreur de la longitude et la latitude. Les tests sont fait aussi dans des conditions semblables pour obtenir des résultats pertinents.

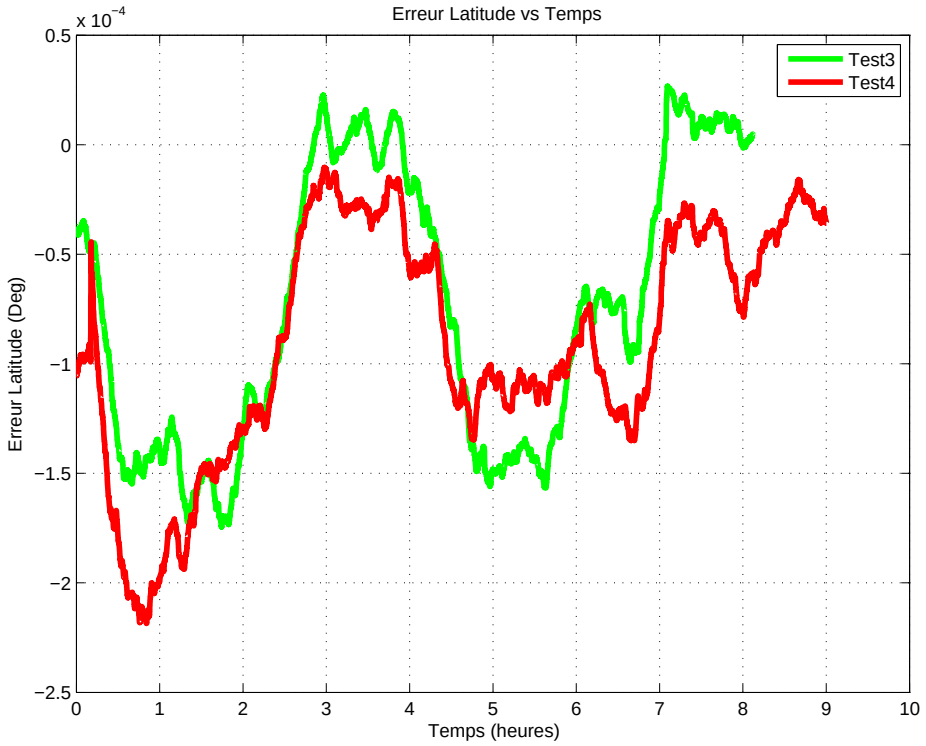


Figure 2.15 Erreur de la latitude pour deux tests

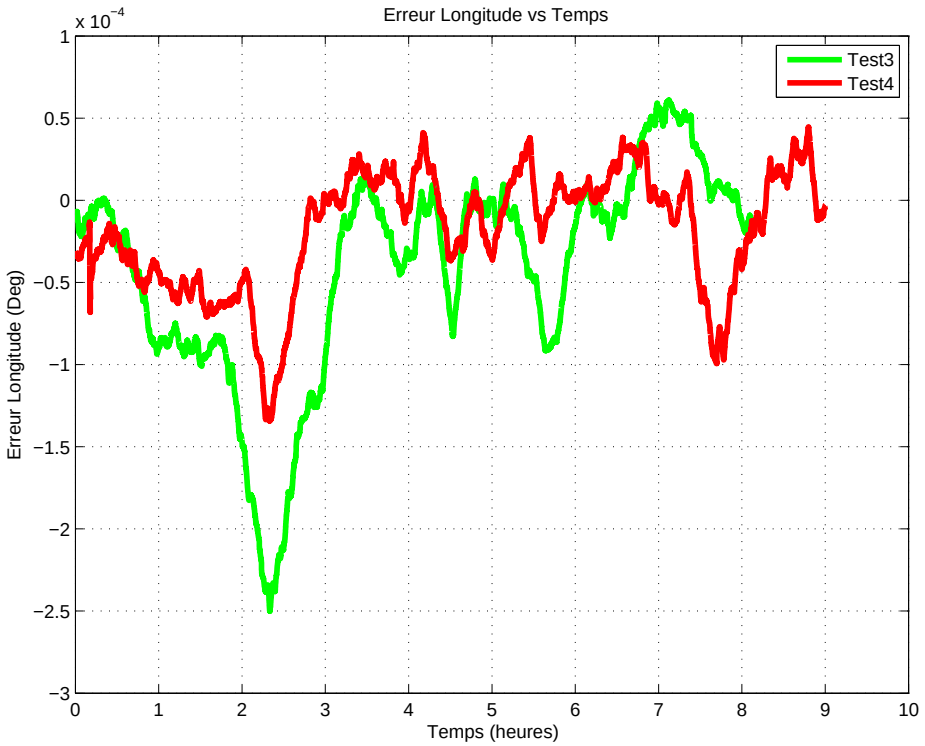


Figure 2.16 l'erreur de la longitude pour deux tests

Figure 2.17 Point du test pour une journée en utilisant le programme GPS visualiser

Les histogrammes ci-dessous montrent que les erreurs en longitude et latitude ne sont pas d'une nature semblable pour chaque jour. Les deux figures sont prise en deux jours différents mais à la même période. Autrement dit, la ressemblance qu'on a montrée n'est pas suffisante pour dire que les erreurs de senseurs GPS se répètent chaque jour, mais ce test est plutôt pour montrer la grande variété des erreurs attendues pendant la navigation sur le trottoir. Cela montre qu'il faut créer un

algorithme pour bien localiser le robot mobile, en fusionnant des données d'odométrie (ou autres capteurs) avec le senseur GPS.

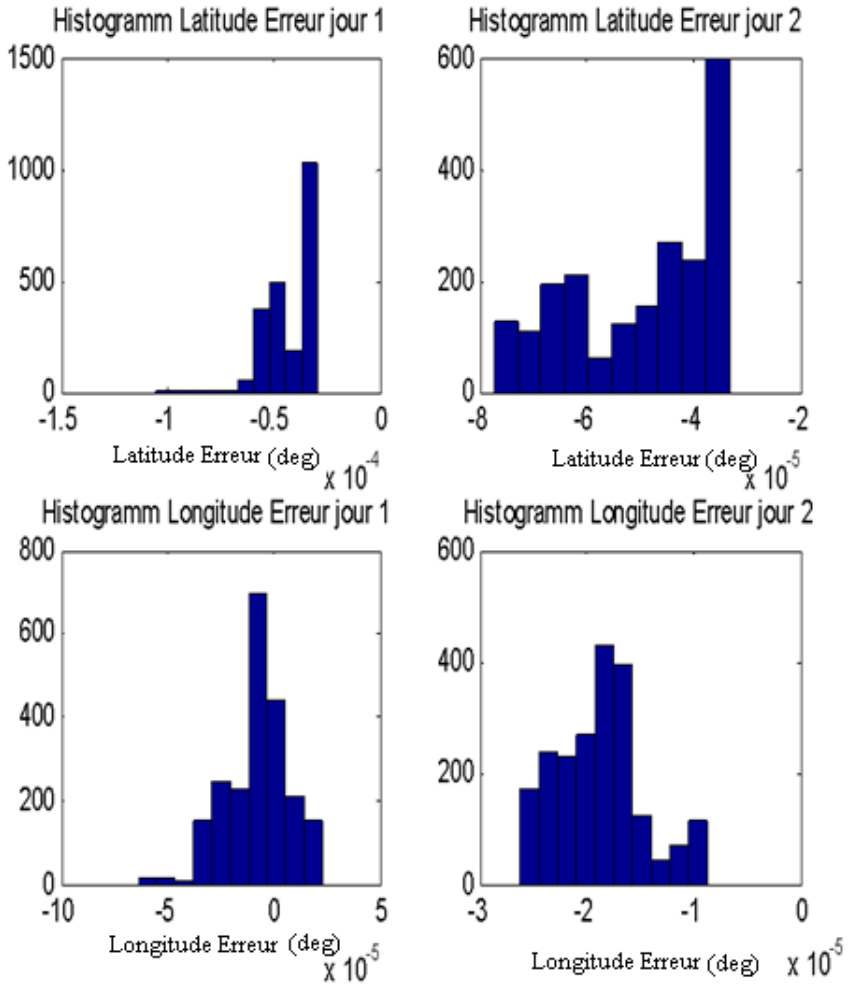


Figure 2.18 Histogrammes de l'erreur sur la longitude et la latitude

2.11 Erreurs de GPS

Les erreurs dans un signal GPS se manifestent sous plusieurs formes et proviennent de plusieurs sources. Les erreurs en général se divisent en deux grandes parties :

1. *bruit de haute fréquence* :
2. *déviation à long terme (Schon, 2006).*

La difficulté de caractériser et de différentier ces erreurs vient de fait que l'erreur « saute » de plusieurs mètres et revient ensuite dans l'autre région où elle se fixe en valeur moyenne pour

quelques secondes avec de faibles erreurs aléatoires, si l'erreur reste fixe pendant 30 seconds, donc elle n'est plus considérée comme un bruit mais plutôt dans la région entre les deux catégories. L'amplitude de ces erreurs peut varier de plusieurs mètres jusqu'à en atteindre les 30 mètres. Les expériences dans ce domaine ont prouvé que les deux raisons principales du bruit de GPS sont l'effet multi-path et les satellites qui entrent et sortent de la liste des satellites vue par le senseur (Kaplan et Hegarty, 2006). La seconde catégorie d'erreurs du GPS est identifiée comme une déviation. Ces erreurs sont plus difficiles à voir sur une figure, étant donné qu'elles changent durant une période de plusieurs heures et pas comme dans le cas précédent. Ce type d'erreurs est attribué à l'effet de l'atmosphère soit dans l'ionosphère ou troposphère et la géométrie du satellite. L'amplitude de ces erreurs peut varier entre une erreur nulle et dix mètres.

2.12 Localisation globale

Ce type de localisation est fait par l'extraction des données latitude et longitude dans le fichier GPX. Ce fichier est généré par l'usager qui doit entrer, par exemple, l'origine comme une intersection et la destination comme une adresse civile. Comme le fichier GPX est écrit en forme de XML, nous avons réaliser les algorithmes pour en extraire les données nécessaires.

2.13 Localisation locale

La localisation locale est faite par le senseur GPS qui calcule la position en tout temps avec une erreur véritable. Le robot fait le contact avec l'environnement par ses capteurs extéroceptifs. La position initiale est calculée à partir des senseurs GPS, et à l'aide de la première position dans le fichier GPX fournie par l'usager.

2.14 Planification de chemin

Il s'agit de calculer la trajectoire à parcourir qui est dans le fichier GPX. D'abord, on doit calculer les distances entre les points, ensuite les orientations entre les segments (le segment est la trajectoire entre deux points consécutifs).

2.15 Longueur de trajet (longueur des segments)

Pour calculer la distance entre deux points sur la terre, on peut utiliser deux équations. La première est la distance grand cercle qui calcule la distance la plus courte sur la terre entre deux points en utilisant la formule de Haversine. Elle suppose que la terre est de forme sphéroïdale en ignorant l'effet ellipsoïdal, ce qui est assez précis pour la plupart des applications (Sinnott, 1984).

En fait, quand Sinnott a conçu la formule de Haversine, la précision numérique était limitée. De nos jours les ordinateurs récents utilisent des représentation de 64 bits à point flottant. Ceci qui nous motive à utiliser la formule de Vincenty, plus précise, de préférence à celle de Haversine.

La formule de Vincenty :

Cette formule (Veness, 2002) utilise un modèle ellipsoïdale de la terre. Elle a deux méthodes itératives utilisées en géodésie. La première méthode (directe) donne la position d'un point par rapport à un autre point en calculant la distance et l'azimut (direction). La deuxième méthode calcule la distance géographique et l'azimut entre deux points donnés. Elles sont largement utilisées grâce à leur précision de 0.5 mm sur l'ellipsoïde de la terre. Supposons que l'on ait deux point $A(Lat1, Lon1)$,$B(Lat2, Lon2)$ où $Lat1$ et $Lat2$ sont les latitudes et $Lon1$,$Lon2$ sont les longitudes des points A,B successivement. On a ainsi

$$\begin{aligned}
\Delta lat &= Lat2 - Lat1 \\
\Delta long &= Long2 - Long1 \\
a &= \sin^2(\Delta Lat/2) + \cos(Lat1).\cos(Lat2).\sin^2(\Delta Long/2) \\
c &= 2.\arctan 2(\sqrt{a}, \sqrt{(1-a)}) \\
d &= R.c \\
R &= 6,371km
\end{aligned}$$

2.16 Éléments d'un trajet

Le trajet est composé de deux éléments qui sont la distance de chaque segment et l'angle entre les segments ; la distance est déjà calculée par les équations précédentes. L'angle de trajet est important pour déterminer l'orientation que le robot doit suivre sur le trottoir. Cette orientation est cruciale parce que la navigation sur le trottoir est limitée par des critères de sécurité et des contraintes de l'environnement. C'est pour cela qu'une petite erreur dans l'orientation pourra engendrer une grande erreur pendant la navigation et les conséquences néfastes sur le robot.

La figure 2.19 illustre les segments et l'orientation d'un trajet. L'orientation est l'angle entre deux segments consécutifs. Si l'orientation est positive, alors le robot va tourner à droite et si elle est négative le robot va tourner à gauche.

2.17 Transposition d'un trajet

Le trajet à parcourir est un décalage d'un trajet déduit des segments AB,BC,CD... (voir figure 2.20). La réduction dépend de la longueur de la rue principale dont on a pris les points A

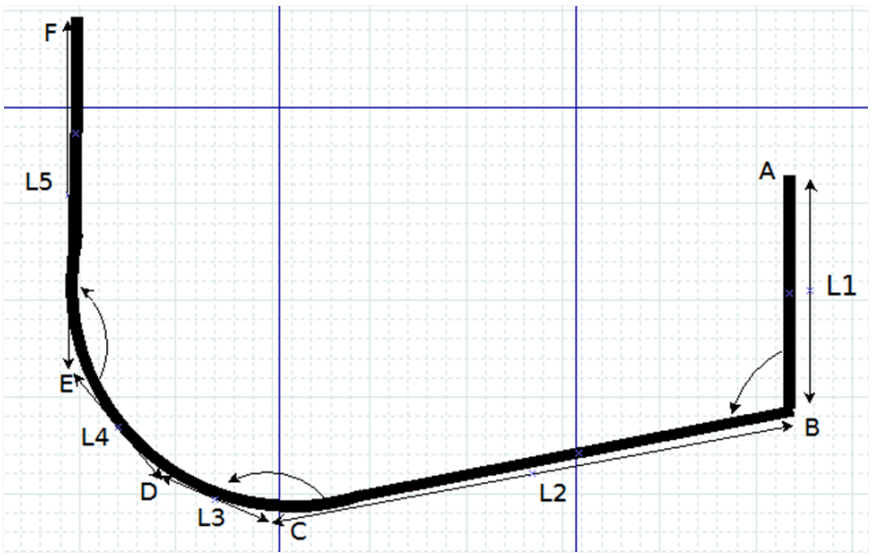


Figure 2.19 Éléments d'un trajet

,B,C,D....

$$\begin{aligned} A_rB_r &= AB - AA_r * \cos(\Theta 1) - BB_r * \cos(\Theta 2) \\ A_rC_r &= BC - BB_r * \sin(\Theta 2) + CC_r * \sin(\Theta 3) \\ C_rD_r &= CD - C_rC * cos(\Theta 3) \end{aligned}$$

La figure 2.20 montre les distances, les orientations et le trajet comme il est dans le fichier GPX ainsi que le trajet réduit. Le trajet de référence à parcourir $A_rB_rC_rD_r$ doit être calculé en ligne. Le premier point de trajet de référence A_r a été inclus dans le biais initial calculé ci-après. Les autres points $B_rC_rD_r$ devront être calculés ou estimés par l'intervention de l'usager ou par l'intermédiaire d'un appareil pouvant distinguer les carrefours durant la navigation.

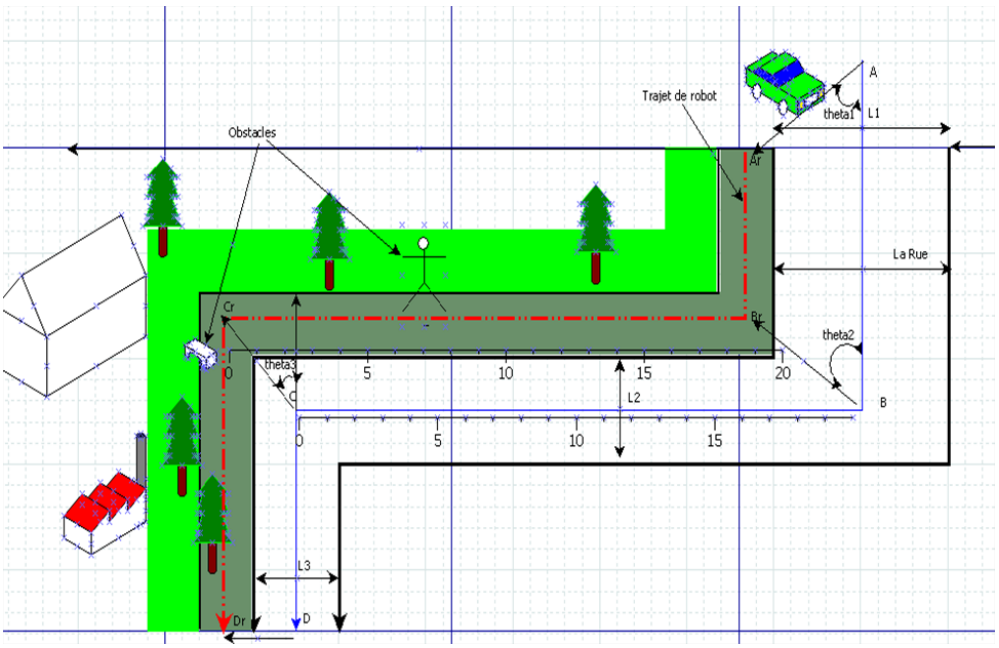


Figure 2.20 Transposition de trajet à faire sur un trottoir

2.18 Repère d'un robot par rapport aux coordonnées GPS

Pour utiliser les coordonnées émises par le senseur GPS, il faut les transférer en coordonnées du robot, autrement dit, les coordonnées de GPS doivent être présentées en forme cartésienne comme c'est le cas en odométrie. Une méthode est suggérée pour faire ces calculs. Les équations suivantes sont nécessaires pour changer de coordonnées géodésiques en données cartésiennes (Featherstone, 1996) :

$$\begin{aligned} X &= (N+h)\cos(\phi)\cos(\lambda) \\ Y &= (N+h)\cos(\phi)\sin(\lambda) \\ Z &= (N(1-e^2)+h)\sin(\phi) \end{aligned}$$

où

- ϕ,λ,h c'est la latitude, la longitude et la hauteur au dessus de l'ellipse.
- X,Y,Z sont les coordonnées terrestre ECEF (Earth-Centered, Earth-Fixed).
- N est le rayon de courbure local à la longitude considérée :

$$N(\phi) = \frac{a}{\sqrt{1-e^2\sin^2(\alpha)}}$$

le rayon de courbure dans la prime verticale.

- a= le demi-grand axe de la terre (le rayon équatorial).
- b= le demi-axe mineur (le rayon polaire).

$$f = \frac{a-b}{a}$$

$$e^2 = 2f - f^2$$

Cette formulation introduit des erreurs à cause des incertitudes avec les mesures de hauteur. Donc, nous devrons prendre une approche que nous avons développée pour projeter les données géodésiques dans le plan local de robot.

2.18.1 Approche de projection de données GPS

Pour projeter la position géodésique d'un point, il faut utiliser deux références (repères). Ces deux références sont nécessaires pour calculer la paire (**Distance**, **Angle**) de ce point par rapport aux références.

Le premier repère est noté R_{dep} admettant pour point de départ (x_{0gps}, y_{0gps}). L'autre est noté R_{fix} admettant pour point d'arrivée (x_{fgps},y_{fgps}) ou son équivalent extrait du fichier GPX. Nous allons considérer que l'erreur moyenne du senseur GPS est fixe pour une période déterminée, chose qui a déjà été illustrée dans les tests expérimentaux faits avec le senseur,

Les étapes pour transformer les coordonnées GPS dans un repère local sont :

1. calculer la distance entre le senseur GPS et le premier repère R_{dep} en utilisant la formule de Vincenty mentionnée ci-dessus ((Veness, 2002)).
2. calculer le relèvement B_{fix} du point de départ (x_{0gps}, y_{0gps}) et le point (x_{fgps}, y_{fgps}) par rapport au Nord (sens horaire).
3. calculer le relèvement $B_s$du point de départ (x_{0gps}, y_{0gps}) et le point du senseur (x_s, y_s).
4. calculer Θ_{tempo} l'angle entre le senseur et l'axe X (en utilisant les deux relèvements précédents (B_{fix},B_s)) en observant plusieurs cas selon la position du senseur par rapport au $Nord$.

Les cas rencontrés pour calculer l'angle Θ_{tempo} sont :

Cas 1 : Le vecteur OB_{fix} est le vecteur entre le point de référence (le dernier point dans le fichier GPX). Le vecteur OX est un vecteur perpendiculaire au vecteur OB_{fix}. Le Θ_{tempo} est l'angle entre le point de senseur GPS par rapport au nouveau repère $B_{fix}OX$. Cette angle est important pour calculer la nouvelle position du senseur dans le repère du robot.

le cas 1 est quand la référence est situé dans le premier quart alors on a : $0 < B_{fix} < 90$

$B_{fix} < B_s$ **&** $B_s > 270 \Rightarrow \Theta_{tempo} = 90 + B_s - B_{fix}$

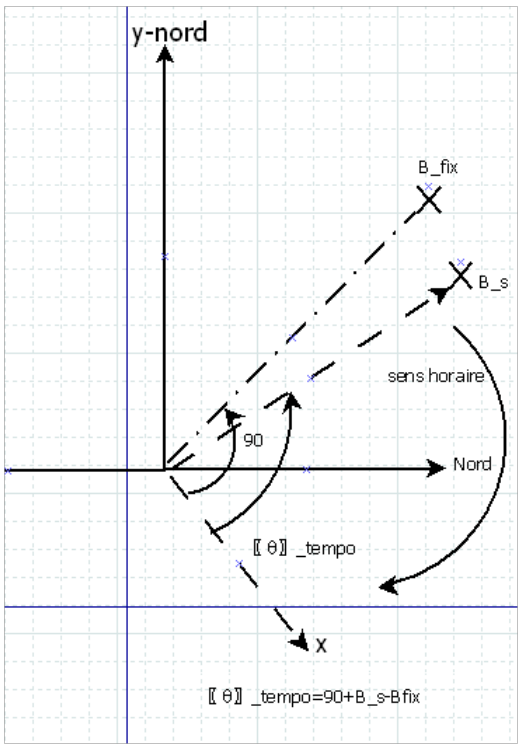


Figure 2.21 Cas 1 où la référence dans le premier quart

Cas 2 : la référence est aussi situé dans le premier quart mais $B_{fix} > B_s$: $0 < B_{fix} < 90$

$$B_{fix} > B_s \Rightarrow \Theta_{tempo} = 90 + abs(B_s - B_{fix})$$

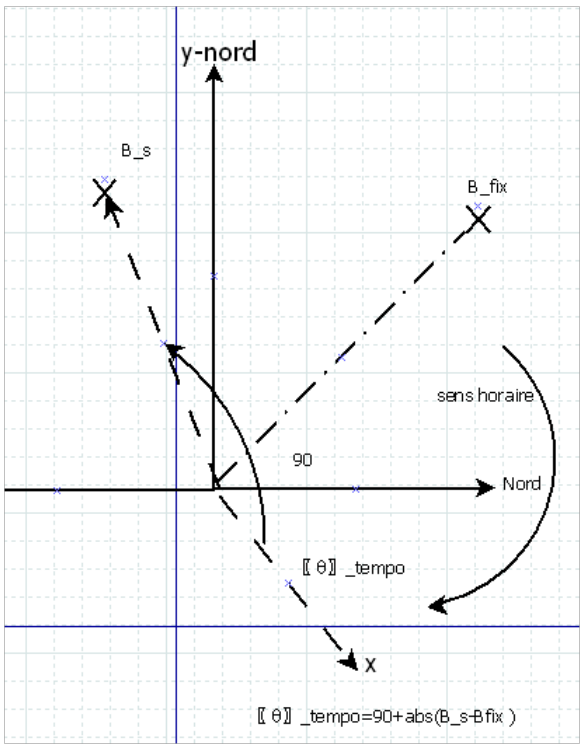


Figure 2.22 Cas 2 où la référence dans le premier quart

Cas 3 : la référence est situé dans le deuxième quart alors on a : $90 < B_{fix} < 180$

$$\Theta_{tempo} = 90 + abs(B_s - B_{fix})$$

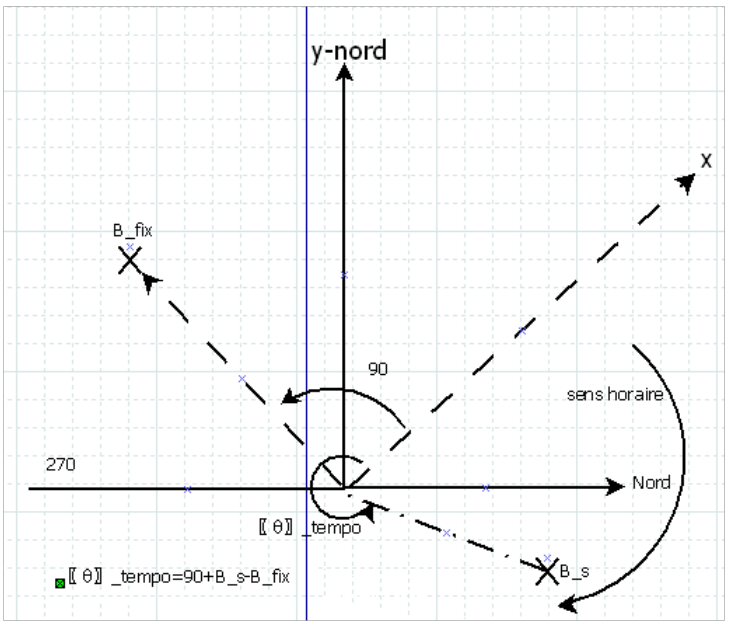


Figure 2.23 Cas 3 où la référence dans le deuxième quart

Cas 4 : la référence est situé dans le troisième quart alors on a : $180 < B_{fix} < 270$

$$\Theta_{tempo} = 90 - (B_{fix} - B_s)$$

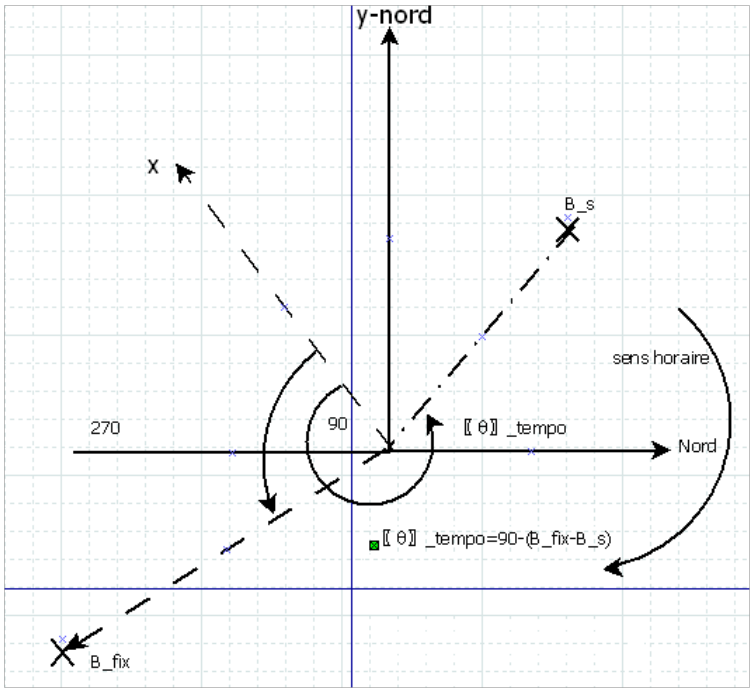


Figure 2.24 Cas 4 où la référence dans le troisième quart

Cas 5 : la référence est situé dans le quatrième quart alors on a : $270 < B_{fix} < 360$

$$\Theta_{tempo} = 90 + B_s - B_{fix}$$

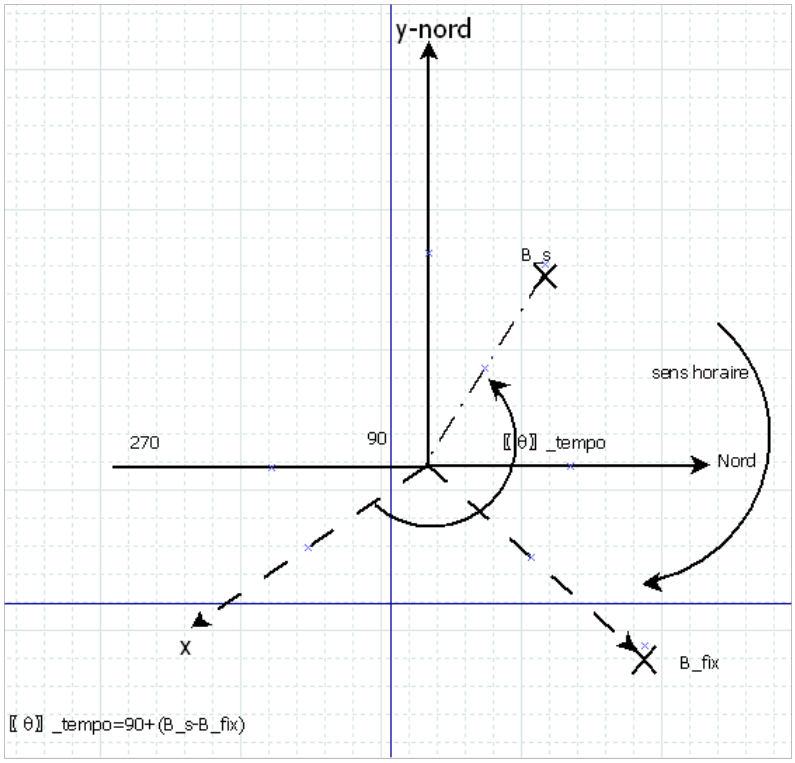


Figure 2.25 Cas 5 où la référence dans le quatrième quart

Après avoir déterminé le cas correspondant à la position du senseur dans les quatre secteurs du cercle. Les cas 1 et 2 sont les cas les plus délicats et souvent rencontrés, car le senseur change son prélèvement autour du Nord. C'est à dire que le changement du prélèvement entre $360°$ et $0°$ change le cas complètement.

2.18.2 Soustraction de biais

Pour minimiser les erreurs de GPS et les ramener au minimum, on considère que le premier point lu par le senseur où le point mesuré pendant l'arrêt (début de course) est le point qui contient les erreurs de biais sur x et y. Alors étant donné que les erreurs du senseur GPS varient d'une façon graduelle et que cette variation prend du temps, on peut, pour les points qui suivent, soustraire ce biais systématiquement. Sachant que cette erreur évolue avec le temps, il parait important de changer le point de biais chaque 30 minutes au plus (une suggestion inspirée par l'expérimentation décrite ci-haut) si la course est faite dans les mêmes conditions ou pas. Ainsi la position du senseur

dans les coordonnés cartésiennes sont :

$$POS_xGps = D_{Gps} * \cos(\Theta_{tempo}) - x_0$$

$$POS_yGps = D_{Gps} * \sin(\Theta_{tempo}) - y_0$$

où le D_{Gps} est la distance entre le senseur et le point de référence (x_{0gps}, y_{0gps}), le (x_0, y_0) est les coordonnées du premier point lu par le senseur GPS.

On remarque sur la figure 2.26 que les distances POS_xGps et POS_yGps sont la projection dans le plan XoY de la distance parcourue D par le robot.

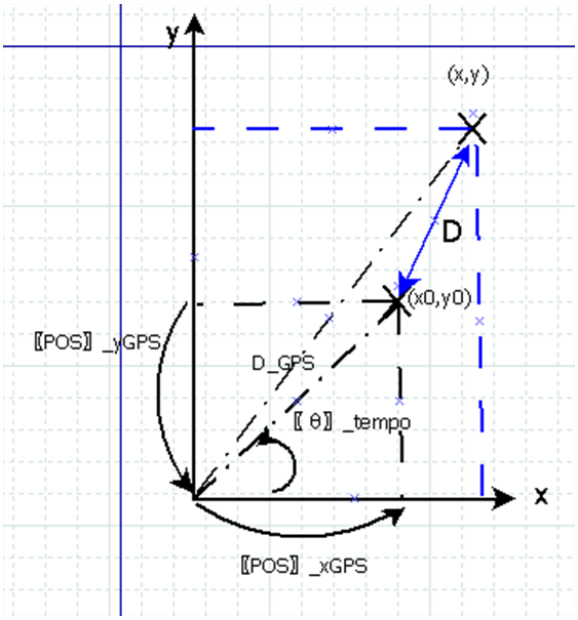


Figure 2.26 Changement de repère du senseur GPS

En repassant sur la figure 2.7, on trouve que un grand biais initial a été enchaîné avec les données GPS. Ce biais revient au fait que ce trajet n'est que un trajet sur le trottoir caché partiellement par les arbres. Dans ce cas, la projection montrée précédemment dans la figure 2.26 peut décaler le trajet vers le droit pour se rapprocher de trajet réel. Ce décalage joue un rôle important dans le processus d'estimation qui, comme on va voir dans la section 3.1.1 de filtre de Kalman, pourrait changer le comportement du filtre. Donc cette projection a des avantages par rapport à la manière dont elle est faite, puisque cette dernière prend le mouvement relatif à partir d'un point de départ et ensuite elle calcule la distance et l'orientation par rapport au point de départ. En soustrayant le décalage initial, on aura une projection locale dans le plan de robot.

2.19 Conclusion

Nous avons présenté dans ce chapitre plusieurs notions liées aux capteurs GPS et leurs applications diverses dans la navigation urbaine. Les tests réalisés avec le capteur illustrent la nature des erreurs du capteur GPS ; une solution ou une méthode a été mise en place pour projeter ces données dans le repère du robot. Le biais dans le capteur GPS pendant un parcours a été supprimé dés le début ; cette idée vient du fait que l'erreur est supposée fixe où bien moins variable pendant une période précise. Les tests expérimentaux ont été faits pendant presque un mois pour de périodes longues pour confirmer ce principe. Les calculs géodésiques étaient nécessaires pour calculer la distance et l'orientation entre deux points sur terre. De plus l'expérimentation nous a donné une idée de comment simuler un GPS lors du travail subséquent. Dans le chapitre suivant, on va proposer une méthode pour simuler un capteur GPS à partir des données odométriques en ajoutant du bruit et un biais. Un petit angle de rotation sera nécessaire car le glissement déforme le trajet de l'odométrie.

CHAPITRE 3

FUSION DE DONNÉES PAR FILTRE DE KALMAN : ESTIMATION DE LA POSITION DU ROBOT

3.1 Filtre de Kalman

3.1.1 Introduction

La navigation met en jeu plusieurs équations mathématiques impliquant des modèles dynamiques issus de la physique du robot et des différents capteurs utilisés. Lorsqu'il s'agit de déterminer certaines variables clés employées pour définir une trajectoire par exemple, il est nécessaire de conjuguer ces modèles ce qui induit des relations mathématiques inextricables et difficilement exploitables. Le filtre de Kalman offre une solution intéressante puisqu'il permet d'abord de fusionner les données collectées indépendamment à partir de chaque modèle puis d'estimer les variables en question. De plus le filtre de Kalman dispose d'une part d'une flexibilité du fait de sa capacité à traiter des équations aussi bien linéaires que non linéaires (par linéarisation) et d'autre part d'optimalité puisqu'il permet de faire l'estimation d'une manière optimale.

3.1.2 Problématique

Le deuxième problème à résoudre apparaît dès qu'il est question d'au moins deux sources de données représentées par deux équations : une équation traduisant la dynamique du robot et une issue du modèle du capteur. Dans notre cas, l'équation dynamique utilise des données provenant de l'odométrie : il s'agit d'informations sur les vitesses angulaire et linéaire ; intégrées pour donner la position et recueillies à partir d'un encodeur incrémental placé sur les roues du robot. Tel qu'il est conçu, le filtre de Kalman prend en compte les équations des sources d'erreurs pour calculer une matrice dites de covariance. Cette dernière a pour rôle d'estimer l'erreur engendrée par chaque cycle de balayage (chaque itération). Les sources d'erreurs sont représentées par une gaussienne ayant un écart type σ. Cette représentation induit la présence d'erreurs accumulées issues de dynamiques cachées et/ou non considérées. Par conséquent, l'odométrie s'en trouve détériorée puisque sensible aux erreurs d'intégration, impliquant par delà des positions erronées. De plus, du fait de sa linéarisation, la dynamique du système lui-même conduit à une estimation imprécise. Dès lors, il est difficile de prédire et le comportement du système et la position du robot exactement. Dans ce chapitre, des méthodes et solutions sont proposées pour améliorer l'intégration de la fusion de données ; moyennant un capteur additionnel moins précis au demeurant.

3.2 Diagramme du filtre de Kalman

Afin de réaliser la meilleur estimation de la position du robot, nous allons recourir au filtre de Kalman. Un schéma représentant les différentes équations du filtre est présentée à la figure 3.1. L'annexe A présente plus en détails ce filtre.

Le processus aléatoire pour l'estimation peut être modélisé comme :

$$x_{(k+1)} = \phi_k x_k + w_k \tag{3.1}$$

L'observation (les mesures) du processus est supposée être produite aux points discrets à la même itération par une relation linéaire :

$$z_k = H_k x_k + v_k \tag{3.2}$$

où

- x_k=$(n*1)$ est le vecteur de processus à l'instantt_k.
- $\phi_k = (n*n)$, la matrice qui relie x_k et $x_(k+1)$en l'absence d'une fonction forcée.
  (si x_k est un échantillon de processus continu, ϕ_k est la matrice d'état de transition usuelle).
- w_k est $(n*1)$, le vecteur de bruit supposé blanc avec une covariance connue.
- $z_k = (m*1)$, le vecteur de mesure à l'instant t_k.
- $H_k = (m*1)$, la matrice donnant la connexion idéale (non bruitée) entre les mesures et le vecteur d'état à l'instant t_k.
- $v_k = (m*1)$, l'erreur des mesures, supposée être une séquence blanche avec une covariance connue et une corrélation nulle avec w_k.

Les matrices de covariance pour les vecteurs v_k et w_k sont données par

$$E(w_k w_i^T) = \left\{ \begin{array}{ll} Q_k & \text{if } i = k \\ 0 & \text{if } i \neq k \end{array} \right\} \tag{3.3}$$

$$E(v_k v_i^T) = \left\{ \begin{array}{ll} R_k & \text{if } i = k \\ 0 & \text{if } i \neq k \end{array} \right\} \tag{3.4}$$

$$E(w_k v_i^T) = 0 \tag{3.5}$$

pour tout k et i.

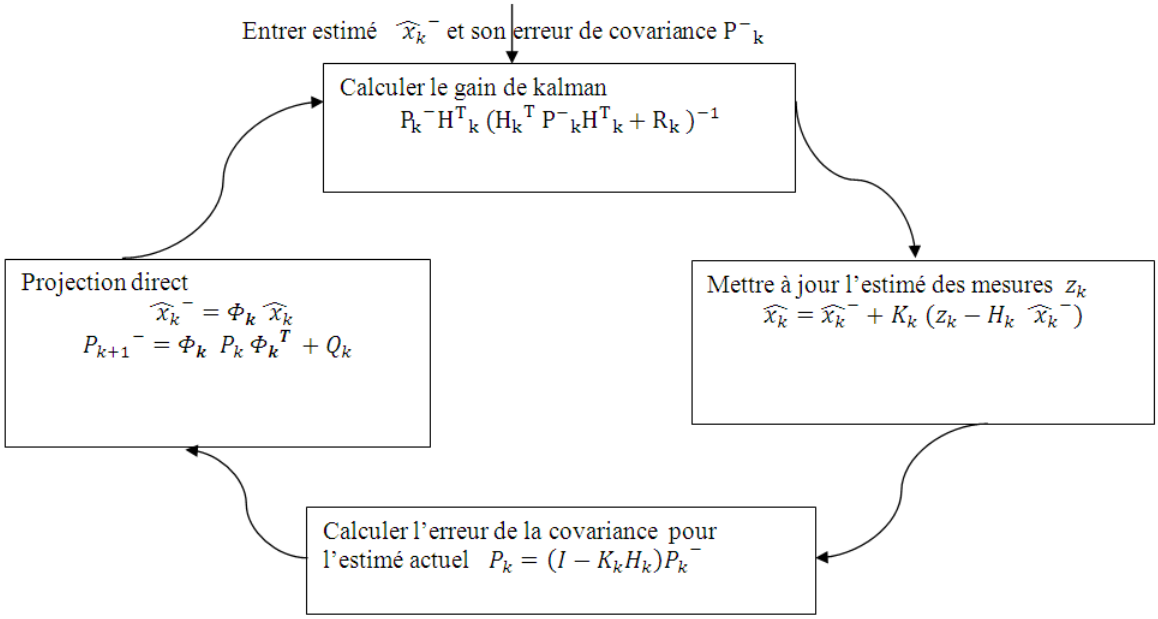


Figure 3.1 Filtre de Kalman (adapté de (De Santis, 2006))

3.3 Module et composants de la cinématique du véhicule

Le robot est représenté par des équations mathématiques extraites des lois de mouvement. Le bloc suivant prend la vitesse linéaire et la vitesse angulaire du véhicule comme entrées et donne la position du robot ainsi que son orientation en sortie.

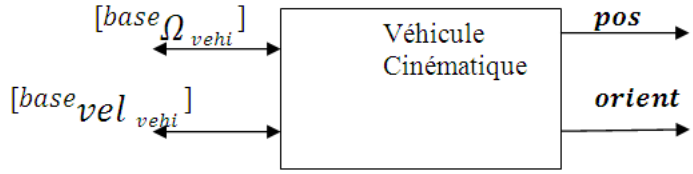


Figure 3.2 Le module cinématique du véhicule (adapté de (De Santis, 2006))

Le module cinématique peut être résumé en deux éléments soit l'intégration et la multiplication qui représentent ses composants essentiels.

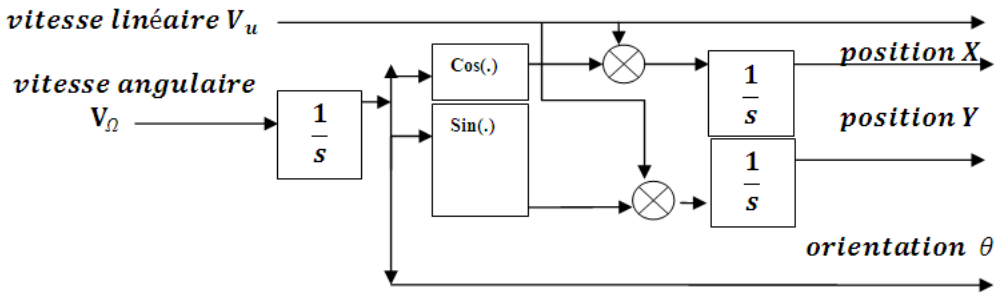


Figure 3.3 Le module cinématique du véhicule en détail (adapté de (De Santis, 2006))

Les équations qui représentent la dynamique du système sont les suivantes :

$$\widehat{x_{k+1}} = \widehat{x_k} + (V_k + w_{v_k}) * \cos(\widehat{\Theta_k}) * \Delta t \tag{3.6}$$

$$\widehat{y_{k+1}} = \widehat{y_k} + (V_k + w_{v_k}) * \sin(\widehat{\Theta_k}) * \Delta t \tag{3.7}$$

$$\widehat{\Theta_{k+1}} = \widehat{\Theta_k} + (\Omega_k + w_{\Omega_k}) * \Delta t \tag{3.8}$$

Le symbole $\widehat{\ }$ (estimé) est utilisé ici pour mettre l'accent sur le fait que les vitesses mesurées sont des vitesses bruitées. Ce bruit est en général modélisé comme étant un bruit blanc.

3.3.1 Module des capteurs

Le bruit associé avec les mesures de vitesses est dû aux capteurs utilisés soit les encodeurs incrémentaux qui emploient la méthode dite de dead reckoning. Cette dernière prend le nombre de rotations des roues pendant une période d'échantillonnage Δt.

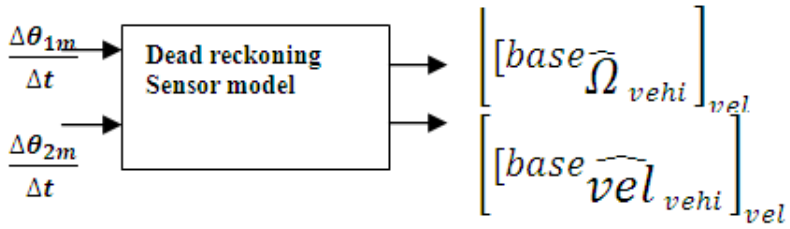


Figure 3.4 Le module des capteurs(adapté de (De Santis, 2006))

Où $\left\{base_{\widehat{\Omega}_{vehi}}\right\}_{vel}$ est l'estimé de la vitesse angulaire des encodeurs optiques :

$$\left\{base_{\widehat{\Omega}_{vehi}}\right\}_{vel} = \frac{\Delta(\theta_1 m)\widehat{r1}(k) - \Delta(\theta_2 m)\widehat{r2}(k)}{\widehat{L}(k)\Delta t}$$

Où $\left\{base_{\widehat{vel}_{vehi}}\right\}_{vel}$ l'estimé de la vitesse linéaire des encodeurs optiques :

$$\left\{base_{\widehat{vel}_{vehi}}\right\}_{vel} = \frac{\Delta(\theta_1 m)\widehat{r1}(k) - \Delta(\theta_2 m)\widehat{r2}(k)}{2\Delta t}$$

Le modèle dynamique de robot est :

$$\widehat{x_{p_{k+1}}} = \widehat{x_{p_k}} + \begin{pmatrix} \widehat{V_k} * \cos(\widehat{\Theta_k}) \\ \widehat{V_k} * \sin(\widehat{\Theta_k}) \\ \widehat{\Omega} \end{pmatrix} \tag{3.9}$$

oú

$$\widehat{x_{p_k}} = \begin{pmatrix} \widehat{x_k} \\ \widehat{y_k} \\ \widehat{\Omega_k} \end{pmatrix}$$

l'estimé de la position, orientation du robot.

3.4 Sources des erreurs

Les estimés du $\widehat{v_k}$ et $\widehat{\Omega_k}$ sont bruités par une erreur :

$$W(k) = \begin{pmatrix} w_{v_k} \\ w_{\Omega_k} \end{pmatrix}$$

Cette erreur provient de plusieurs sources. $\widehat{V_k} = V_k + w_{v_k}$et $\widehat{\Omega_k}$=$\Omega_k + w_{\Omega_k}$ sont les vitesses estimés à partir des encodeurs optiques. Les équations de l'odométrie avec des vitesses estimées sont les suivantes :

$$\widehat{x_{k+1}} = \widehat{x_k} + (V_k + w_{v_k}) * \cos(\widehat{\Theta_k}) * \Delta t \tag{3.10}$$

$$\widehat{y_{k+1}} = \widehat{y_k} + (V_k + w_{v_k}) * \sin(\widehat{\Theta_k}) * \Delta t \tag{3.11}$$

$$\widehat{\Theta_{k+1}} = \widehat{\Theta_k} + (\Omega_k + w_{\Omega_k}) * \Delta t \tag{3.12}$$

en simplifiant les trois équations précédentes on aura :

$$\widehat{x_{k+1}} = \widehat{x_k} + V_k * \cos(\widehat{\Theta_k}) * \Delta t + w_{v_k} * \cos(\widehat{\Theta_k}) * \Delta t \quad (3.13)$$

$$\widehat{y_{k+1}} = \widehat{y_k} + V_k * \sin(\widehat{\Theta_k}) * \Delta t + w_{v_k} * \sin(\widehat{\Theta_k}) * \Delta t \quad (3.14)$$

$$\widehat{\Theta_{k+1}} = \widehat{\Theta_k} + \Omega_k * \Delta t + w_{\Omega_k} * \Delta t \quad (3.15)$$

On trouve que les équations précédentes peuvent être représentées par la forme générale :
$X_{p_{k+1}} = X_{p_k} + B_k + G_k * W_k$

oú

$$\widehat{X_{p_k}} = \begin{pmatrix} \widehat{x_k} \\ \widehat{y_k} \\ \widehat{\Omega_k} \end{pmatrix}, \widehat{B_k} = \begin{pmatrix} 0 & 0 & V_k * \cos(\widehat{\Theta_k}) * \Delta t \\ 0 & 0 & V_k * \sin(\widehat{\Theta_k}) * \Delta t \\ 0 & 0 & w_{\Omega_k} * \Delta t \end{pmatrix},$$

$$\widehat{G_k} = \begin{pmatrix} \Delta t * \cos(\widehat{\Theta_k}) & 0 \\ \Delta t * \sin(\widehat{\Theta_k}) & 0 \\ 0 & \Delta t \end{pmatrix}, \widehat{X_{p_{k+1}}} = \begin{pmatrix} \widehat{x_{k+1}} \\ \widehat{y_{k+1}} \\ \widehat{\Omega_{k+1}} \end{pmatrix},$$

$$\widehat{w_k} = \begin{pmatrix} w_{v_k} \\ w_{\Omega_k} \end{pmatrix}$$

Les équations de l'odométrie sans erreur sont les suivantes :

$$x_{k+1} = x_k + V_k * \cos(\Theta_k) * \Delta t \quad (3.16)$$

$$y_{k+1} = y_k + V_k * \sin(\Theta_k) * \Delta t \quad (3.17)$$

$$\Theta_{k+1} = \Theta_k + \Omega_k * \Delta t \quad (3.18)$$

Les équations de l'erreur sont les suivantes :

$$\tilde{x_{k+1}} = x_{k+1} - \widehat{x_{k+1}} \quad (3.19)$$

$$\tilde{y_{k+1}} = y_{k+1} - \widehat{y_{k+1}} \quad (3.20)$$

$$\tilde{\Theta_{k+1}} = \Theta_{k+1} - \widehat{\Theta_{k+1}} \quad (3.21)$$

De l'équation 3.19

$$\tilde{x_{k+1}} = x_k - \widehat{x_k} + V_k * \cos(\Theta_k) * \Delta t - V_k * \cos(\widehat{\Theta_k}) * \Delta t - w_{v_k} * \cos(\widehat{\Theta_k}) * \Delta t$$

$$= \tilde{x_k} + V_k * (\cos(\Theta_k) - \cos(\widehat{\Theta_k})) * \Delta t - w_{v_k} * \cos(\widehat{\Theta_k}) * \Delta t \quad (3.22)$$

Oú

$$\begin{aligned} cos(\Theta_k) - cos(\widehat{\Theta_k}) &= -2\sin(\frac{\Theta_k - \widehat{\Theta_k}}{2}) * \sin(\frac{\Theta_k + \widehat{\Theta_k}}{2}) \\ &= -2\sin(\frac{\Theta_k + \widehat{\Theta_k}}{2}) * \frac{\tilde{\Theta_k}}{2} \\ &= -\sin(\frac{\Theta_k + \widehat{\Theta_k}}{2}) * \tilde{\Theta_k} \end{aligned} \tag{3.23}$$

$$\begin{aligned} \sin(\frac{\Theta_k + \widehat{\Theta_k}}{2}) &= \sin(\frac{2 * \Theta_k + \tilde{\Theta_k}}{2}) \\ &= \sin(\widehat{\Theta_k} + \frac{\tilde{\Theta_k}}{2}) \\ &= \sin(\widehat{\Theta_k}) * \cos(\frac{\tilde{\Theta_k}}{2}) + \cos(\widehat{\Theta_k}) * \sin(\frac{\tilde{\Theta_k}}{2}) \end{aligned} \tag{3.24}$$

en considérant $\widehat{\Theta_k}$ très petit, alors $\cos(\widehat{\Theta_k}) = 1$ et $\sin(\widehat{\Theta_k}) = \widehat{\Theta_k}$

$$\sin(\frac{\Theta_k + \widehat{\Theta_k}}{2}) = \sin(\widehat{\Theta_k}) + \cos(\widehat{\Theta_k}) * \frac{\tilde{\Theta_k}}{2} \tag{3.25}$$

En substituant l'équation 3.25 à l'équation 3.23 on aura :

$$\begin{aligned} cos(\Theta_k) - cos(\widehat{\Theta_k}) &= -\sin(\widehat{\Theta_k}) * \tilde{\Theta_k} - \cos(\widehat{\Theta_k}) * \frac{\tilde{\Theta_k^2}}{2} \\ &= \sin(\widehat{\Theta_k} + \frac{\tilde{\Theta_k}}{2}) \\ &= \sin(\widehat{\Theta_k}) * \cos(\frac{\tilde{\Theta_k}}{2}) + \cos(\widehat{\Theta_k}) * \sin(\frac{\tilde{\Theta_k}}{2}) \end{aligned} \tag{3.26}$$

l'équation 3.19 de l'erreur sur x devient :

$$\tilde{x_{k+1}} = \tilde{x_k} - \widehat{V_k} * \sin(\widehat{\Theta_k}) * \tilde{\Theta_k} * \Delta t - w_{v_k} * \cos(\widehat{\Theta_k}) * \Delta t \tag{3.27}$$

En faisant la même chose pour l'équation 3.20 :

$$\begin{aligned} \tilde{y}_{k+1} &= y_k - \widehat{y_k} + V_k * \sin(\Theta_k) * \Delta t - \widehat{V_k} * \sin(\widehat{\Theta_k}) * \Delta t - w_{v_k} * \sin(\widehat{\Theta_k}) * \Delta t \\ &= \tilde{y}_k + \widehat{V_k} * (\sin(\Theta_k) - \sin(\widehat{\Theta_k})) * \Delta t - w_{v_k} * \sin(\widehat{\Theta_k}) * \Delta t \end{aligned} \tag{3.28}$$

oú

$$\begin{aligned} sin(\Theta_k) - sin(\widehat{\Theta_k}) &= 2\cos(\frac{\Theta_k + \widehat{\Theta_k}}{2}) * \sin(\frac{\Theta_k - \widehat{\Theta_k}}{2}) \\ &= -\tilde{\Theta_k} * cos(\frac{\Theta_k + \widehat{\Theta_k}}{2}) \end{aligned} \tag{3.29}$$

En simplifiant $cos(\frac{\Theta_k + \widehat{\Theta_k}}{2})$ on obtient :

$$\begin{aligned} \cos(\frac{\Theta_k + \widehat{\Theta_k}}{2}) &= y\cos(\widehat{\Theta_k} + \frac{\tilde{\Theta_k}}{2}) \\ &= \cos(\tilde{\Theta_k}) * \sin(\frac{\tilde{\Theta_k}}{2}) + \cos(\widehat{\Theta_k}) * \sin(\frac{\tilde{\Theta_k}}{2}) \end{aligned} \tag{3.30}$$

En prenant les mêmes considérations pour l'angle $\widehat{\Theta_k}$:

$$\cos(\frac{\Theta_k + \widehat{\Theta_k}}{2}) = \cos(\widehat{\Theta_k}) - \sin(\widehat{\Theta_k}) * \frac{\tilde{\Theta_k}}{2} \tag{3.31}$$

on substitue l'équation 3.31 à l'équation 3.29 :

$$\begin{aligned} sin(\Theta_k) - sin(\widehat{\Theta_k}) &= \cos(\widehat{\Theta_k}) * \tilde{\Theta_k} - \sin(\widehat{\Theta_k}) * \frac{\tilde{\Theta_k^2}}{2} \\ &= \cos(\widehat{\Theta_k}) * \tilde{\Theta_k} \end{aligned} \tag{3.32}$$

En remplaçant l'équation 3.32 à l'équation 3.28

$$\tilde{y_{k+1}} = \tilde{y_k} + \widehat{V_k} * \cos(\widehat{\Theta_k}) * \tilde{\Theta_k} * \Delta t - w_{v_k} * \sin(\widehat{\Theta_k}) * \Delta t \tag{3.33}$$

Et de la même manière l'équation 3.21 devient :

$$\tilde{\Theta_{k+1}} = \tilde{\Theta_k} + (\widehat{\Omega_k} - \Omega_k) * \Delta t + w_{\Omega_k} * \Delta t \tag{3.34}$$

Alors le modèle d'erreur :

$$\begin{pmatrix} \xi_{x_{k+1}} \\ \xi_{y_{k+1}} \\ \xi_{\Theta_{k+1}} \end{pmatrix} = A(k) * \begin{pmatrix} \xi_{x_k} \\ \xi_{y_k} \\ \xi_{\Theta_k} \end{pmatrix} + G(k) * w(k) \tag{3.35}$$

oú

$$A_k = \begin{pmatrix} 1 & 0 & V_k * \sin(\widehat{\Theta_k}) * \Delta t \\ 0 & 1 & V_k * \cos(\widehat{\Theta_k}) * \Delta t \\ 0 & 0 & 1 \end{pmatrix}, G_k = \begin{pmatrix} -\Delta t * \cos(\widehat{\Theta_k}) * \Delta t & 0 \\ -\Delta t * \sin(\widehat{\Theta_k}) * \Delta t & 0 \\ 0 & -\Delta t \end{pmatrix}$$

Et la matrice de covariance P

$$P_{p_{k+1}} = A_k * P_k * A_k^{'} - G_k * Q_k * G_k^{'}$$

Oú

$$\widehat{Q_k} = \begin{pmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\Omega^2 \end{pmatrix}$$

et σ_v,σ_Ω sont les écarts types de la vitesse linéaire et angulaire successivement.

3.5 Intégration des données de l'odométrie avec les données GPS

Le capteur GPS fournit la position par rapport à un repère connu, par conséquent on a :

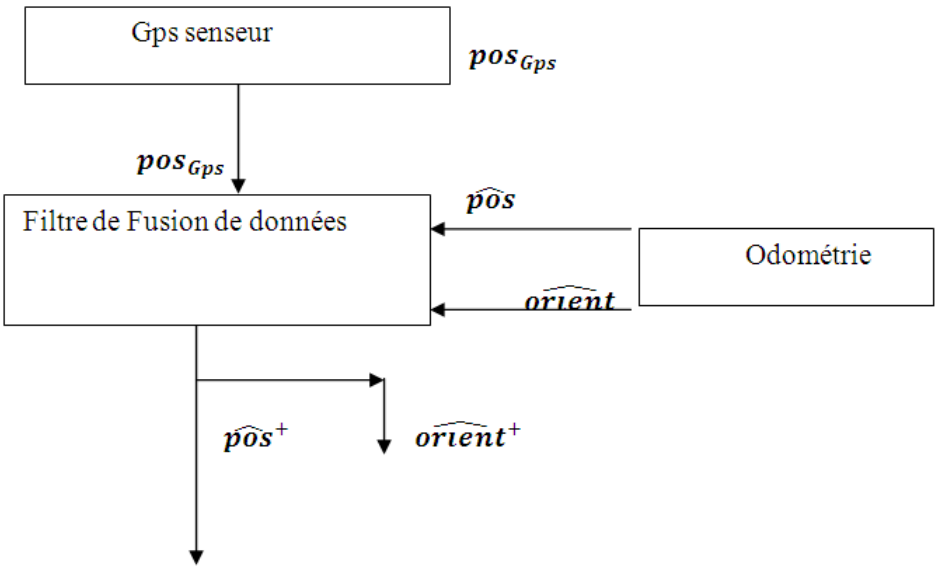


Figure 3.5 Fusion des données (adapté de (De Santis, 2006))

Le diagramme précédent illustre la fusion de données entre deux sources.

3.6 Gain du filtre et la mise à jour de la covariance

$$S_{k+1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} * P_k * \begin{pmatrix} 1 & \grave{0} & 0 \\ 0 & 1 & 0 \end{pmatrix} + R$$

Le gain :

$$K_{k+1} = P_{p_{k+1}} * \begin{pmatrix} 1 & \grave{0} & 0 \\ 0 & 1 & 0 \end{pmatrix} * S_{k+1}^{-1}$$

Et la mise à jour de la matrice P :
$P_{k+1} = P_{p_{k+1}} - K_{k+1} * Sk + 1 * K_{k+1}^{'}$

3.7 Localisation par le filtre de Kalman

Le robot se localise moyennant deux méthodes complémentaires, la première étant l'odométrie, la deuxième étant basée sur la mesure du capteur GPS. L'odométrie donne la position dans un plan XoY avec une erreur accumulée. Cette dernière est élevée eu égard à l'orientation, Surtout lorsqu'il s'agit de grandes distances parcourues induisant de grandes erreurs d'intégration. Le capteur GPS quant à lui manifeste des erreurs élevées en début de parcours et vis-à-vis des courtes distances parcourues. Ces erreurs liées soit à l'odométrie et au capteur GPS sont inhérentes aux capteurs eux mêmes et ne peuvent être surmontées par des techniques logicielles. La solution adoptée ci-après propose de combiner les performances des deux méthodes via une procédure de fusion de données basée sur un algorithme accordant plus de poids à telle ou telle mesure des capteurs selon la plage de distance dans laquelle se trouve le robot soit courte ou longue distance. Ainsi au démarrage et jusqu'à un certain point qui sera défini expérimentalement l'algorithme de fusion de données pondèrera fortement les mesures provenant de l'odométrie. Au-delà de ce point, l'algorithme donnera plus de poids aux mesures émanant du capteur GPS.

3.7.1 Navigation à courte distance

Modèle dynamique de l'odométrie représentant le modèle réel du robot.

Dans la navigation à courte distance, le GPS sera seulement utilisé à des fins de confirmation. Ainsi l'information transmise par ce dernier sera juste employée pour donner une idée quant à la position du robot, et non comme mesures nécessaires à la navigation ; car à ce stade l'odométrie est plus précise. Par simulation, la trajectoire estimée est conforme à la trajectoire de l'odométrie, car la variance de l'odométrie est plus petite que celle du GPS. Alors pendant le processus d'estimation, le filtre du Kalman suit toujours l'odométrie peu importe le bruit inhérent au GPS. On remarque dans la figure 3.6 que les trois trajectoires sont superposées sur la même figure, la trajectoire du GPS est

introduite en simulation en bruitant les données de l'odométrie à la manière d'une gaussienne de moyenne nulle et d'écart type relativement grand.

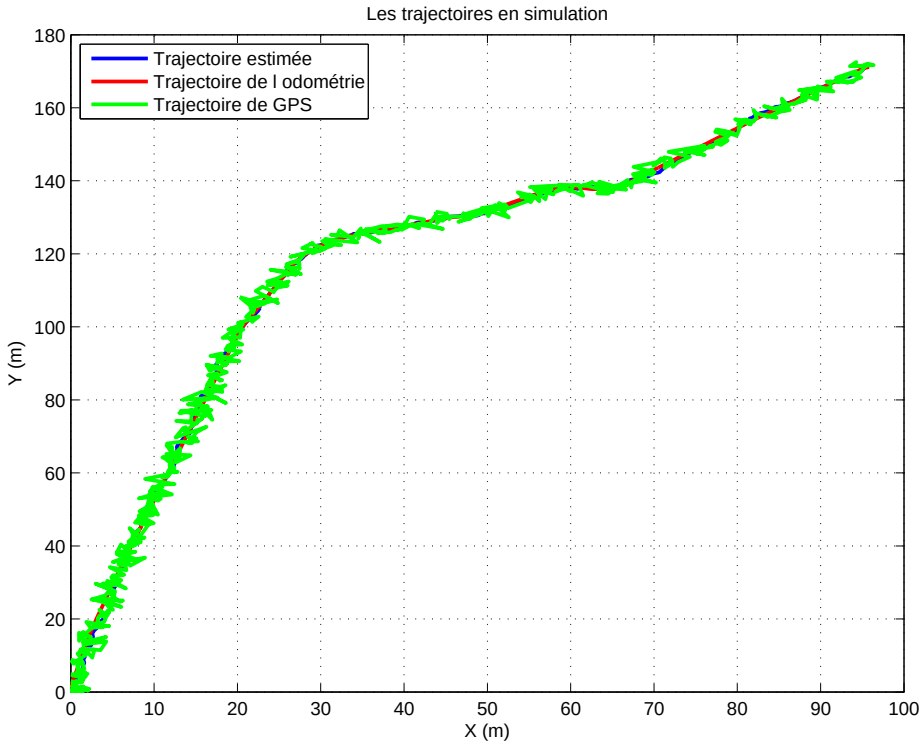


Figure 3.6 Trajectoires estimée, odométrie, et GPS

En traçant la diagonale de la matrice de covariance P du filtre, on remarque que la covariance commence avec une valeur initiale quelconque et ensuite diminue avec une grande pente. L'oscillation entre deux valeurs constantes se répète pendant toute la trajectoire suivie.

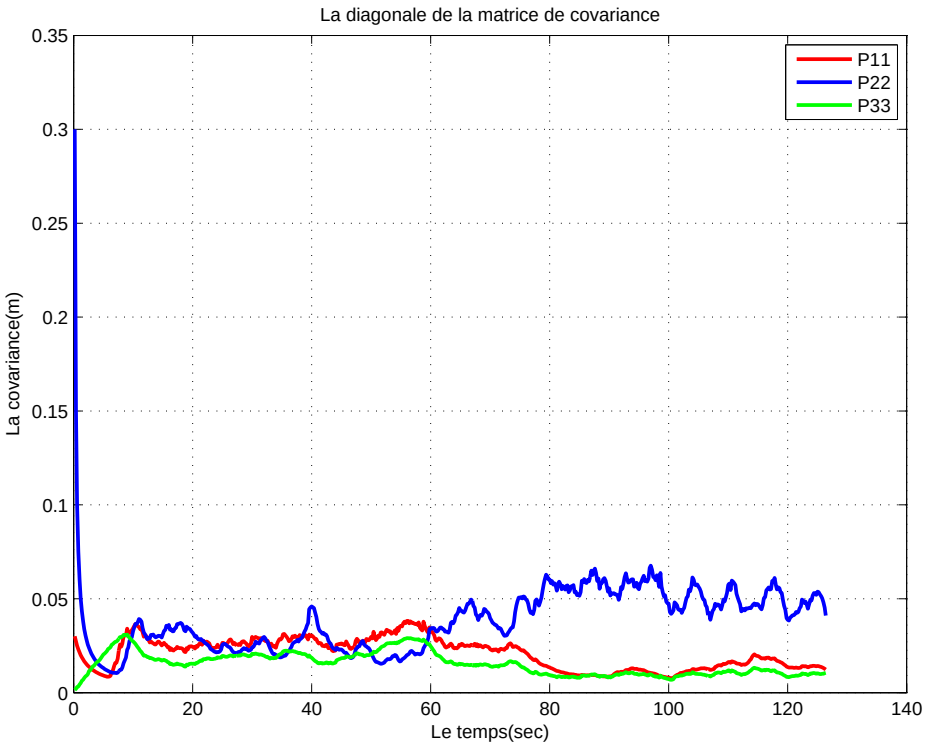


Figure 3.7 la matrice de covariance P du filtre

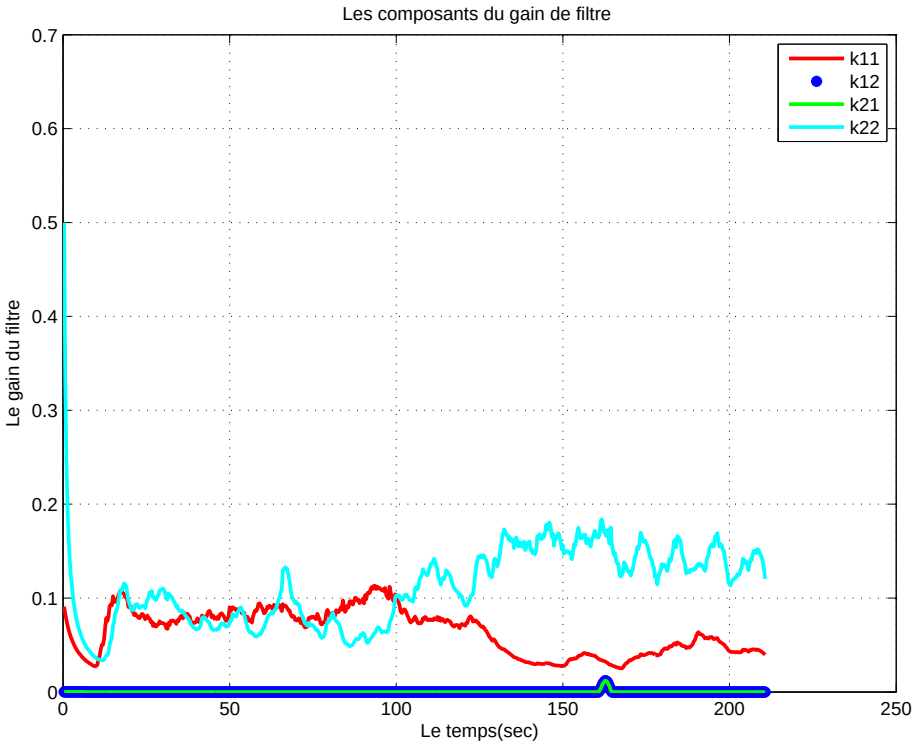


Figure 3.8 le gain du filtre

On remarque que le gain diminue d'une manière semblable à celle de la matrice de covariance, néanmoins les gains $k12$ et $k21$ sont symétriques.

Modèle dynamique de l'odométrie ne représentant pas fidèlement le modèle réel du robot

L'odométrie donne la position du robot suite à une intégration de la vitesse linéaire et angulaire en utilisant les équations qui représentent la dynamique du système. Pourtant la dynamique de l'erreur ne correspond pas à celle calculée dans les équations du filtre, car le bruit est ajouté juste à la vitesse linéaire et angulaire, cela n'est pas suffisant en présence de glissement et le robot utilisé est fait pour tourner même quand il glisse.

3.7.2 Navigation à longue distance

Le capteur GPS sera utilisé pour naviguer durant les longues distances et effectuer le test de kidnapping du robot. Le test de kidnapping repose sur l'idée de déplacer le robot par un moyen de transport auxiliaire en prenant soin d'arrêter la navigation durant le déplacement. La navigation redémarre au moment où le robot reprend son autonomie soit au moment du débarquement de ce dernier. En guise d'exemple d'application considérons un usager utilisant une chaise roulante à navigation autonome, si cet utilisateur doit prendre le bus la chaise roulante cesse de fonctionner ainsi que son algorithme de navigation, néanmoins cette dernière doit redémarrer et continuer à fonctionner correctement au moment où l'usager devra descendre du bus. Étant donné que le robot est en déplacement, l'odométrie lors de ce test ne sera pas capable de déterminer la position de ce dernier ; d'où le rôle du capteur GPS qui sera lui en revanche en mesure de donner une position correcte au robot au moment où il doit se réamorcer. Toutefois il est à noter que le robot utilisera l'information GPS seulement pour restituer sa position puisqu'il s'agit en vérité d'un nouveau démarrage et que par conséquent le relais devra être donné à l'odométrie. Cependant, il est important de réinitialiser la navigation par odométrie sinon cette dernière sera accompagnée d'erreurs accumulées lors de la précédente session de navigation. Ainsi le plugin responsable de la mise à jour des points de navigation, à savoir le NavPointToPoint (voir l'annexe B pour la description de l'architecture de commande), sera lancé au moment du chargement de l'algorithme de navigation et ses paramètres.

3.7.3 GPS en simulation

Les mesures de l'odométrie sont utilisées comme source de données pour générer les données du GPS en simulation. Ainsi en introduisant un bruit blanc d'une moyenne nulle et d'écart type relativement grand à ces données d'odométrie, on peut prétendre à un simulateur de GPS. Expérimentalement les erreurs de GPS (non différentiel) sont de type apériodiques et biaisées avec une précision variable pouvant atteindre les 30 mètres d'erreur. L'application de la méthode suggérée pour localiser les données GPS dans le repère du robot dénote qu'il y a une rotation de trajectoire

du GPS par rapport à la trajectoire réelle. Le calcul approximatif de cet angle de rotation se fait en ligne au début de la trajectoire. Ce faisant, on peut néanmoins dire que le gabarit de la trajectoire à suivre reste correct. Pour simuler une trajectoire de GPS, on prend la trajectoire de l'odométrie bruitée à laquelle on ajoute un autre bruit modélisé par une gaussienne de moyenne nulle et d'écart type assez grand. Ensuite on effectue une rotation instantanée de chaque point du GPS. La simulation dans la figure 3.6 est faite avec un GPS simulé avec et un angle de rotation.

3.7.4 Fusion de données du GPS

Le module de la fusion de données du filtre de Kalman est décrit dans les équations du filtre. Ce modèle a pour but de représenter le processus de fusion de données, et nous incite à nous poser des questions de base relatives à ce sujet à savoir

- Pourquoi la fusion ?
- Que doit-on fusionner ?
- Comment effectuer cette fusion ?
- Est-ce vraiment utile de fusionner ?

Dans le cadre de ce travail, ces questions d'ordre général peuvent être traduites de la manière suivante :

- Quelle est la nature des données à fusionner (numériques/symboliques, continues/discrètes, certaines/incertaines, temporelles/non-temporelles, ...) ?
- Quelles sont les propriétés des processus à modéliser (observables, partiellement observables, markoviens, non-markoviens, ...) ?
- Quelles sont les exigences en termes de résultats (sensibilité, spécificité du système) ?

Pour nous, la fusion de données se produit entre le GPS soit un capteur qui donne une position bruitée du robot la position accumulée par l'odométrie qui est calculée à partir des équations de la dynamique du robot. Pour que la fusion de données soit opérationnelle, on doit distinguer deux éventualités. La première implique la mise en association de deux appareils de même ordre de précision ; dans ce cas il est nécessaire que les écarts types des deux appareils soient de même nature avec des valeurs proches. La deuxième éventualité implique quant à elle l'utilisation de deux appareils ayant des ordres de précision différents, c'est-à-dire que l'un est beaucoup plus précis que l'autre. Il est à noter que du fait de contraintes de coût par exemple liées à l'emploi de l'appareil précis, on conçoit que ce dernier ne sera utilisé qu'occasionnellement. En vue de balayer tous les cas, et dans le cadre d'une étude comparative on a essayé de concevoir un GPS en simulation avec plusieurs valeurs d'écart type. Pour l'odométrie, il a été associé un écart type fixe soit σ_V=0.0022 ; Ω_V=0.0135.

À la figure 3.9, la trajectoire estimée suit la trajectoire du GPS. Conformément à la théorie de Kalman, le filtre est capable de rejeter le bruit en éliminant les oscillations liées aux données

brutes du GPS. En vue de justifier pourquoi l'estimé suit le GPS il faut regarder l'erreur à chaque cycle d'estimation. Autrement dit, il faut regarder la diagonale de la matrice de covariance P. La figure 3.10 montre les trois éléments de la matrice de covariance. Au début, les valeurs initiales sont assez grandes, puis elles diminuent à chaque cycle d'estimation donnant une covariance fixe à la fin. Une covariance oscillant entre deux valeurs maximale et minimale ; garde un estimé des trajectoires selon un comportement fixe soit en suivant le trajet de l'odométrie ou bien le trajet du GPS. La comparaison des deux figures (figure de covariance et figure des gains du filtre) révèle que les gains du filtre de Kalman coïncident avec leurs correspondants en termes de gabarit. Cela est justifié par le fait que le filtre génère une commande basée sur la covariance qui elle est reliée à l'élément correspondant. On remarque aussi que les K_{12}, K_{21} sont des éléments symétriques. Donc on peut dire en générale que le gain du filtre de Kalman n'est qu'une image de la matrice de covariance à une différente échelle.

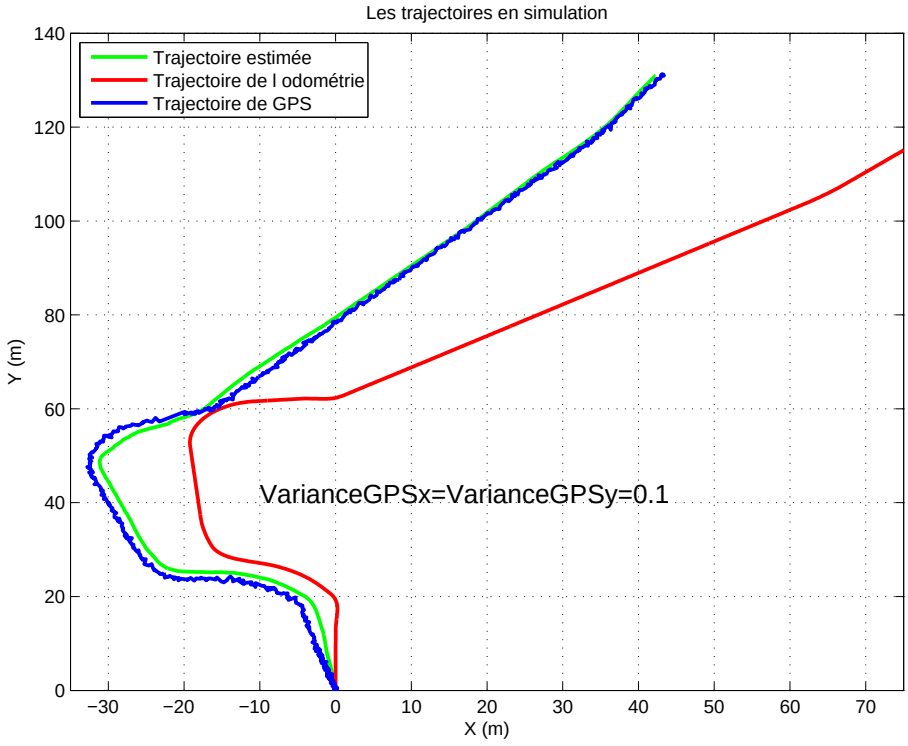
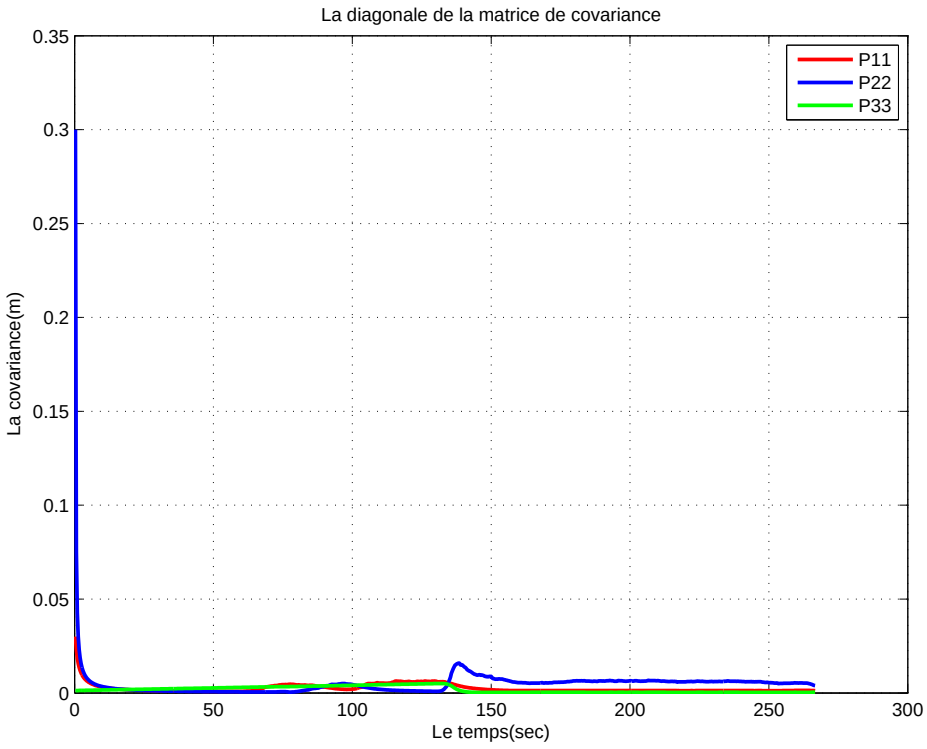


Figure 3.9 Trajectoires simulées d'un parcours fait par le robot avec un GPS en simulation

(a)



(b)

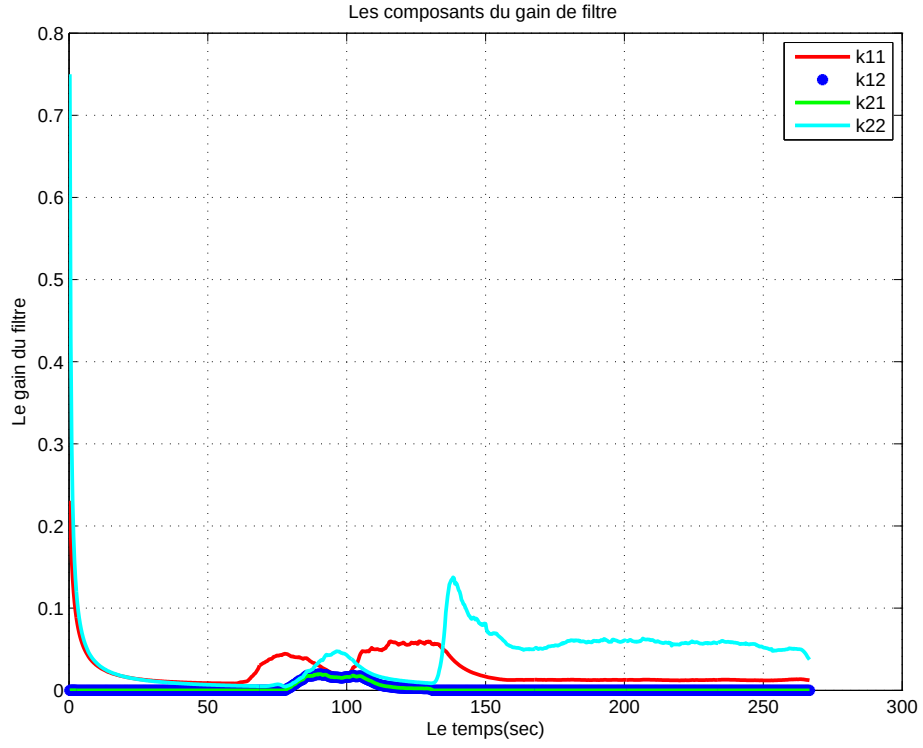


Figure 3.10 Diagonale de la matrice de covariance et le gain de filtre correspondant

En augmentant l'écart type du GPS (figure 3.11), l'estimé se rapproche de plus en plus de la trajectoire de l'odométrie. Le filtre élimine le bruit du GPS et garde en même temps une position encadrée par les limites des courbes de trajectoires GPS et odométrie. Cependant la courbe de la trajectoire de l'estimé est cette fois ci plus proche de celle de l'odométrie qu'elle ne l'était lors des tests précédents. Ceci s'explique par le fait que le parcours soit long ce qui induit une grande covariance. Après 40 mètres, la trajectoire de l'estimé suit l'orientation de celle du GPS bien qu'elle reste proche de la trajectoire de l'odométrie.

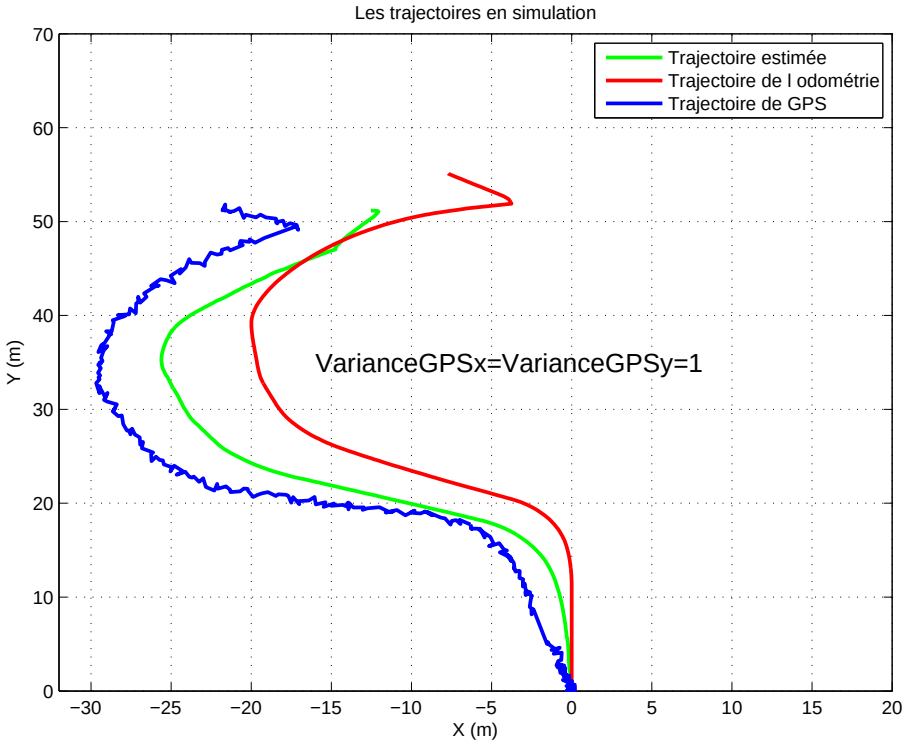
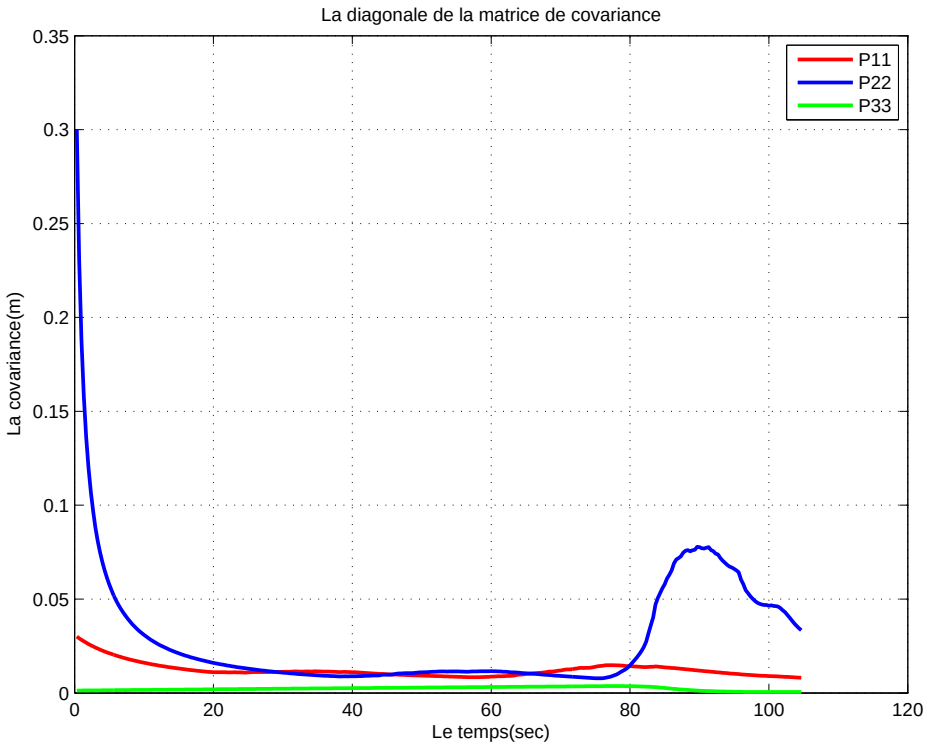


Figure 3.11 Trajectoires simulées d'un parcours fait par le robot avec un GPS en simulation

(a)



(b)

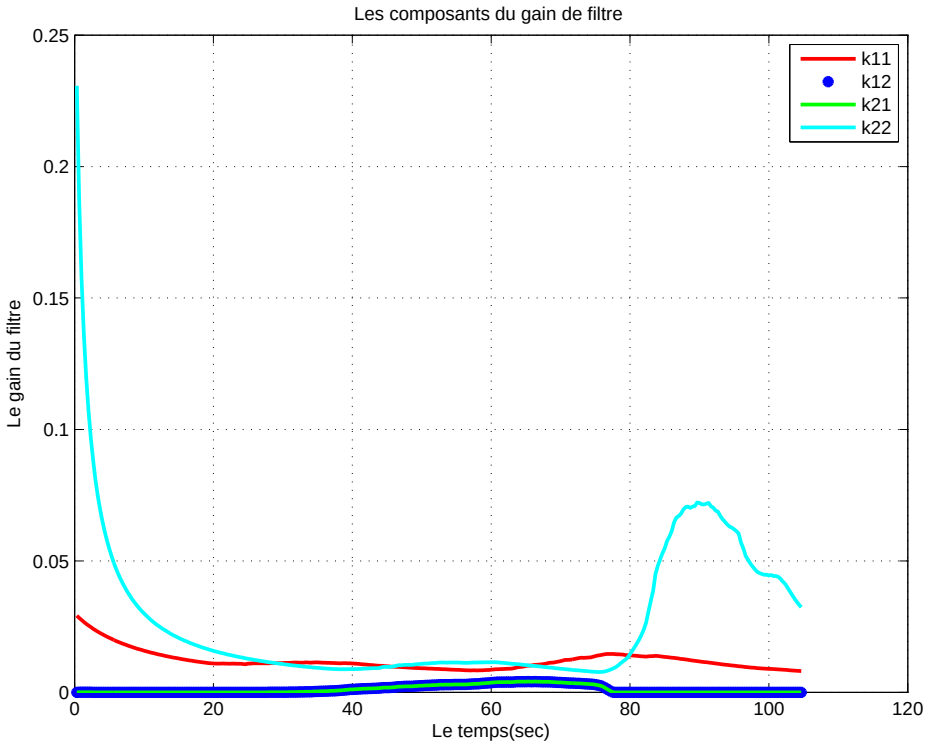


Figure 3.12 Diagonale de la matrice de covariance et le gain de filtre correspondant

En augmentant l'écart type du GPS, l'estimé se rapproche de plus en plus de la trajectoire de l'odométrie. Le filtre élimine le bruit du GPS et garde en même temps une position encadrée par les limites des courbes de trajectoires GPS et odométrie. Cependant la courbe de la trajectoire de l'estimée est cette fois ci plus proche de celle de l'odométrie qu'elle ne l'était lors des tests précédents. Ceci s'explique par le fait que le parcours soit long ce qui induit une grande covariance. Après 40 mètres, la trajectoire de l'estimé suit l'orientation de celle du GPS bien qu'elle reste proche de la trajectoire de l'odométrie. La figure 3.13 illustre ce cas.

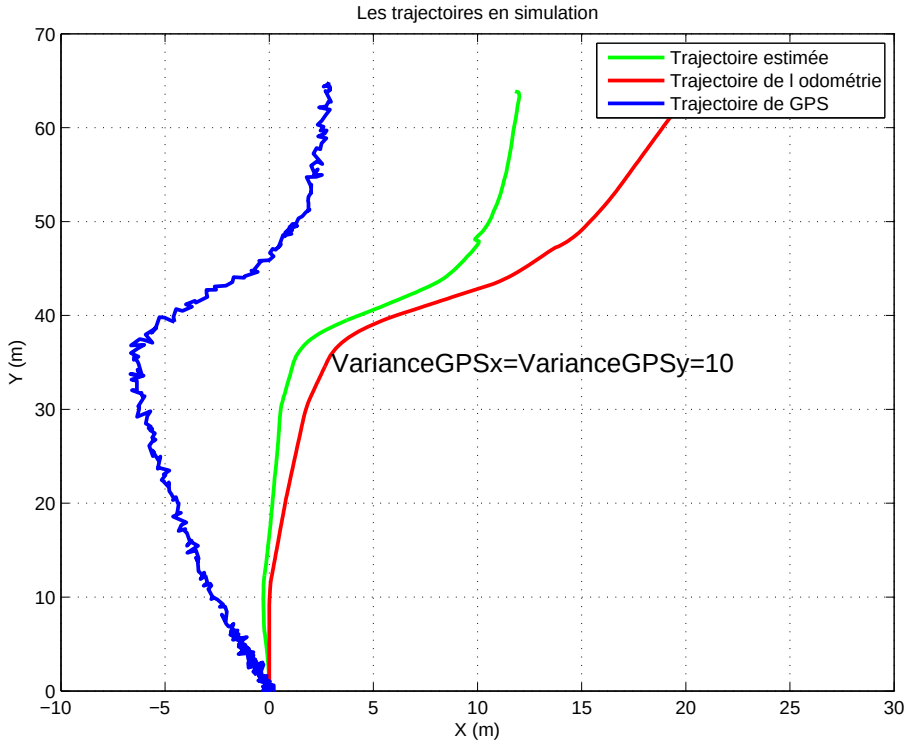
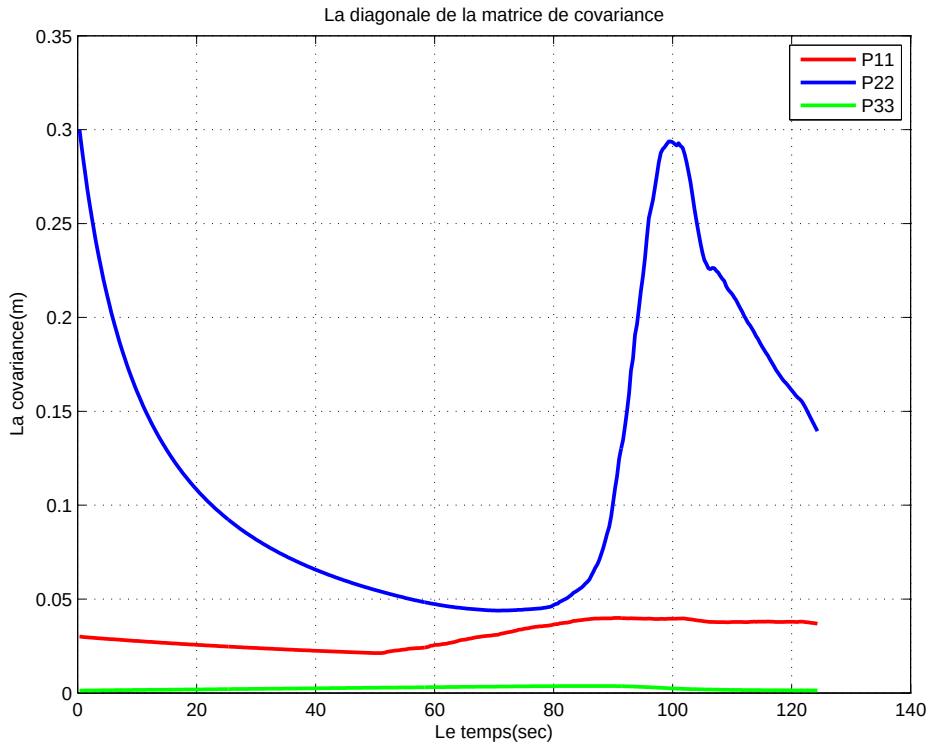


Figure 3.13 Trajectoires simulées d'un parcours fait par le robot avec un GPS en simulation

(a)



(b)

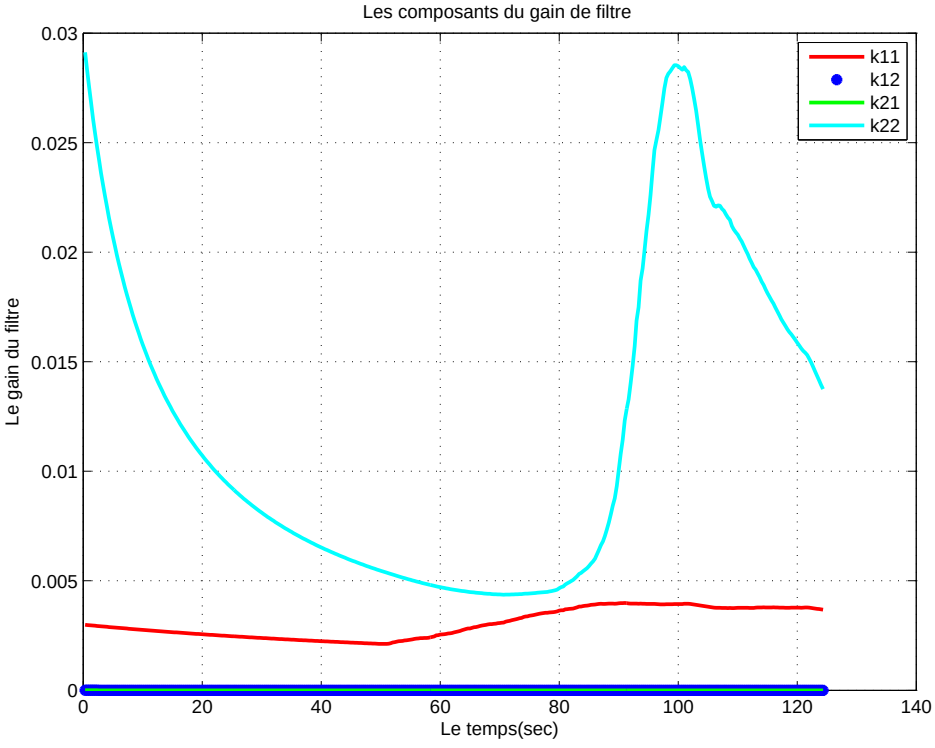


Figure 3.14 Diagonale de la matrice de covariance et le gain de filtre correspondant

La figure 3.14 décrit une courbe convexe, elle montre en l'occurrence la présence d'un sommet durant le processus, cela est dû à l'éloignement entre les deux trajectoires GPS et odométrie. Cela étant dit, on voit, par la suite, que l'estimé suit l'odométrie pour diminuer la covariance.

3.8 Résultats expérimentaux

La partie expérimentale, déroulée en trois étapes, vient répondre aux questions suivantes :

- Le robot est il capable de suivre un trajet complexe impliquant plusieurs points GPS dans une zone urbaine ?
- Le robot est il capable de parcourir la trajectoire désirée et d'effectuer des taches connexes à savoir l'évitement d'obstacles ainsi que la détection des bords du trottoir ?
- Le robot peut il faire un suivi de trajectoire sous forme de ligne droite en l'absence d'obstacles puis revenir au point de départ avec une bonne précision ?
- Quelle est l'erreur de différence relevée entre le chemin parcouru et la trajectoire désirée ?
- Si l'on disposait d'un GPS avec une précision de moins de 3 m lorsqu'il y a présence d'un système de correction d'erreur W.A.A.S, alors le robot peut il effectuer un suivi de trajectoire satisfaisant dans des quartiers urbains ?

La navigation à l'extérieur est une tâche difficile et complexe du fait de la conjugaison de plusieurs facteurs environnementaux (obstacles, nature du sol, climat etc.) et de l'association de plusieurs appareils et instruments (senseurs, correcteurs, actionneurs etc.). C'est pour cela qu'il faut calibrer tous les dispositifs séparément en vue de mieux cerner les limites de chacun, ceci afin que l'analyse de résolution de problèmes ultérieures soit optimale.

En effet, dans notre cas nous avons commencé dans un premier temps dans le cadre d'un test préliminaire (test 1) par calibrer le GPS en vérifiant les données retournées par ce dernier lors d'une navigation sans utiliser le robot.

Le deuxième test consiste en le suivi d'une trajectoire rectangulaire en mode manuel soit avec l'utilisation du Joystick comme moyen de contrôle. Ceci afin de comparer les résultats issus de la fusion des données du GPS de l'odométrie c'est-à-dire la trajectoire estimée avec les trajectoires réel et désirée. De plus, ce test permet de vérifier la concordance des coordonnées GPS projetées avec les coordonnées géodésiques.

Le troisième test vise à examiner le bon fonctionnement du plugin NavPointToPoint responsable de l'émission des commandes linéaires et angulaires. Ainsi, on substitue le plugin en question au joystick, basculant dès lors du mode manuel vers un mode assisté (semi-autonome). Dans le cadre de ce test, hormis des paramètres liés à l'initialisation et le fonctionnement du plugin, aussi bien la trajectoire que tous les autres paramètres de navigation restent identiques à ceux du test 2. Il est à noter que la trajectoire rectangulaire est censée reproduire des conditions de navigation réalistes

puisqu'elle met en jeu des notions de boucle, de retour à un point de départ, etc.

Le quatrième test a pour but de suivre fidèlement une ligne droite en vue de reconstituer des conditions de navigation sur un trottoir par exemple. Ainsi, hormis ce changement de circuit à parcourir, il est sensiblement identique au test précédent.

La cinquième et dernière expérimentation consiste en l'introduction de toutes les conditions de navigation réelles dans un test où les situations de présence d'obstacles et de dénivellations sont rencontrées. La trajectoire parcourue reste une ligne droite et le mode de fonctionnement adopté est le mode semi-autonome.

Il faut savoir que le climat renforce la difficulté liée à ce type de navigation (à l'extérieur), puisque le développement d'un algorithme efficace implique beaucoup de tests d'essais-erreurs qui devraient idéalement se produire sous les mêmes conditions. En effet, plus il sera effectué de tests et plus les calibrages et autres ajustements de paramètres seront exacts La complexité liée à la réalisation de tests à l'extérieur nous oblige à introduire des notions relatives au repérage dans l'espace et aux différentes coordonnées géographiques. Ainsi, une attention particulière est à accorder lors des projections des trajectoires dans le plan local censées refléter la réalité.

L'expérimentation donne une idée sur la fiabilité des données issues de l'odométrie et des erreurs liées à la dynamique du système. Les résultats provenant de ces expérimentations indiquent que pour simuler une fusion de données avec un appareil GPS de manière satisfaisante et fiable, il faut d'abord partir de données odométriques comme source de données du GPS simulé. Par la suite, il faut les bruiter. Dans un second plan, il faudra aussi bruiter la vitesse angulaire afin de simuler l'effet du frottement.

Il est aussi à noter que les tests ont été réalisés dans divers endroits, pour minimiser l'influence du sol et des autres facteurs environnementaux pouvant agir sur le frottement.

3.8.1 Test numéro 1

Après avoir appliqué l'hypothèse mentionnée dans la section 2.10 qui stipule que l'erreur du capteur GPS est fixe pendant une période déterminée et que le premier point est la référence temporelle pour naviguer, le biais embarqué dans les donnés du GPS a pu être éliminé. La figure 3.15 montre la validité de cette hypothèse et prouve par voie expérimentale qu'il y un angle de rotation à corriger pendant les premiers pas du trajet ; La trajectoire parcourue lors de ce test est rectangulaire (tour d'une résidence). Ce test représente une zone urbaine, puisque le trottoir est caché par des arbres. Le signal GPS est bruité, par le double trajet. Cet angle est égal à -17° degrés et est calculé par la pente de la tangente des premiers points du trajet (par rapport au Nord) et la verticale 90°. En faisant une rotation approximative égale à cet angle, on trouve un trajet conforme à celui généré graphiquement par le programme *GPSvisualiser* (Schneider, 2002). Dans la projection, une rotation de cet angle est nécessaire pour que le trajet soit conforme à l'axe Y car le plan dans lequel le

robot évolue n'est pas encore défini. Par conséquent à chaque fois qu'on change le point de départ on aura un angle de rotation différent. Dans tous les cas, le segment de départ doit être conforme à l'axe Y dans le repère du robot. Dans les figures 3.15 , 3.16, on peut vérifier l'exactitude de la projection dans Matlab et dans la carte géodésique.

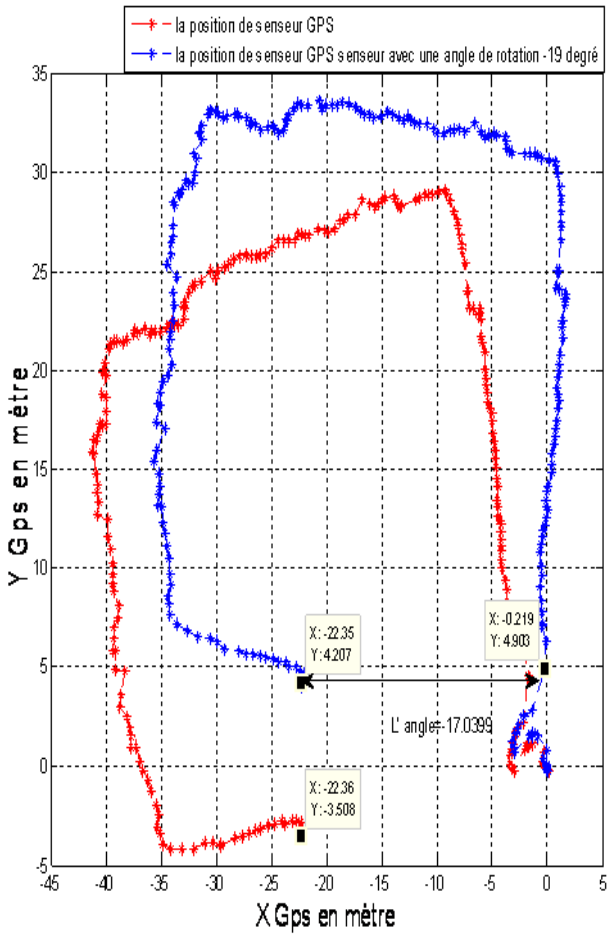


Figure 3.15 Trajet sous forme d'un carré

Figure 3.16 Trajet sur une carte géodésique

3.8.2 Test numéro 2

Dans le test 2, on enclenche le mode manuel ; utilisant ainsi le joystick comme moyen de contrôle ; en vue de réaliser le suivi de trajectoire désirée. Le trajet désiré est un carré avec un périmètre de 122.3m. On constate, sur la figure 3.17, que lorsque l'on utilise le joystick, l'odométrie délivre une trajectoire moins déformée que celle que l'on va rencontrer dans le test suivant impliquant le plugin NavPointToPoint. Cette moindre déformation de la trajectoire de l'odométrie est issue du fait que le joystick (commandé par l'usager) donne les vitesses angulaires et linéaires simultanément. L'usager devient dans ce cas le compensateur appliquant si nécessaire et instantanément les corrections requises assurant le suivi de la trajectoire désirée. Autrement dit, lorsque l'on donne les vitesses angulaires et linéaires pour effectuer un virage, abordant ce dernier graduellement, le robot tourne avec un angle proche de celui demandé. Ce qui n'est pas le cas avec le plugin NavPointToPoint. Puisque ce dernier donne les vitesses indépendamment. De plus le frottement n'est pas modélisé dans la conception de ce plugin. Dans les figures 3.19 et 3.20, on a pris des points sur le trajet carré ; le relevé des ces points a été fait par un technicien en génie civil. On a pris un point de référence dans le parking C de l'École Polytechnique pour calculer la position des points du trajet désiré.

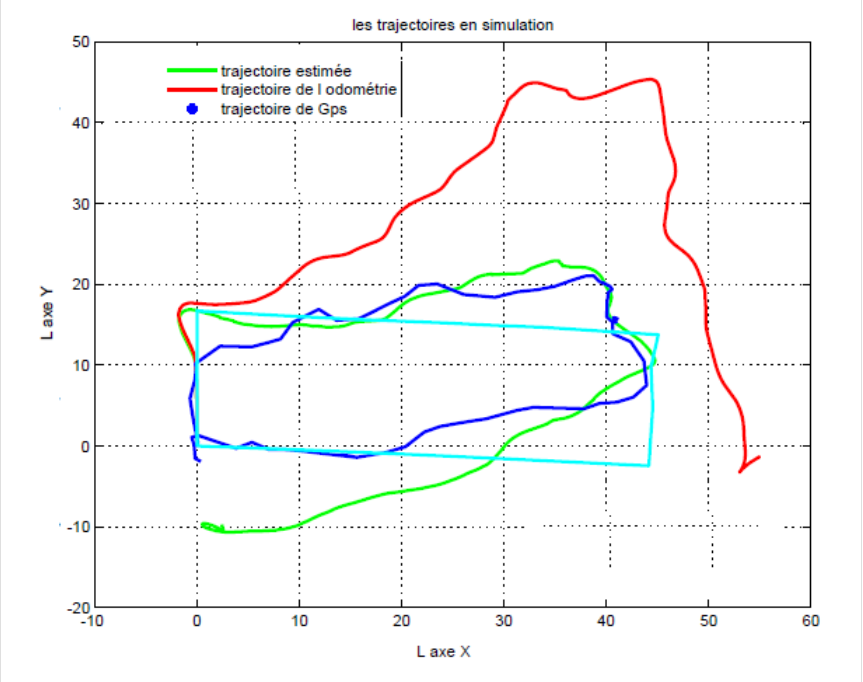


Figure 3.17 Trajet sous forme d'un carré

Figure 3.18 Trajet sur une carte géodésique

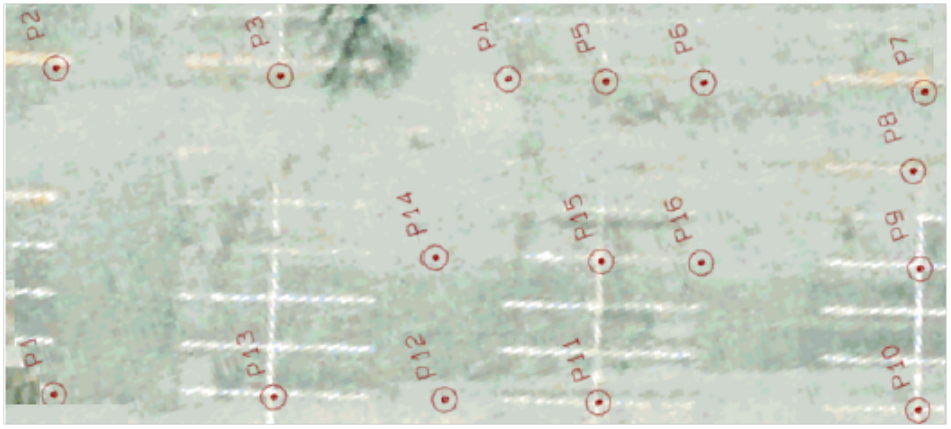


Figure 3.19 Points du trajet sur une carte géodésique

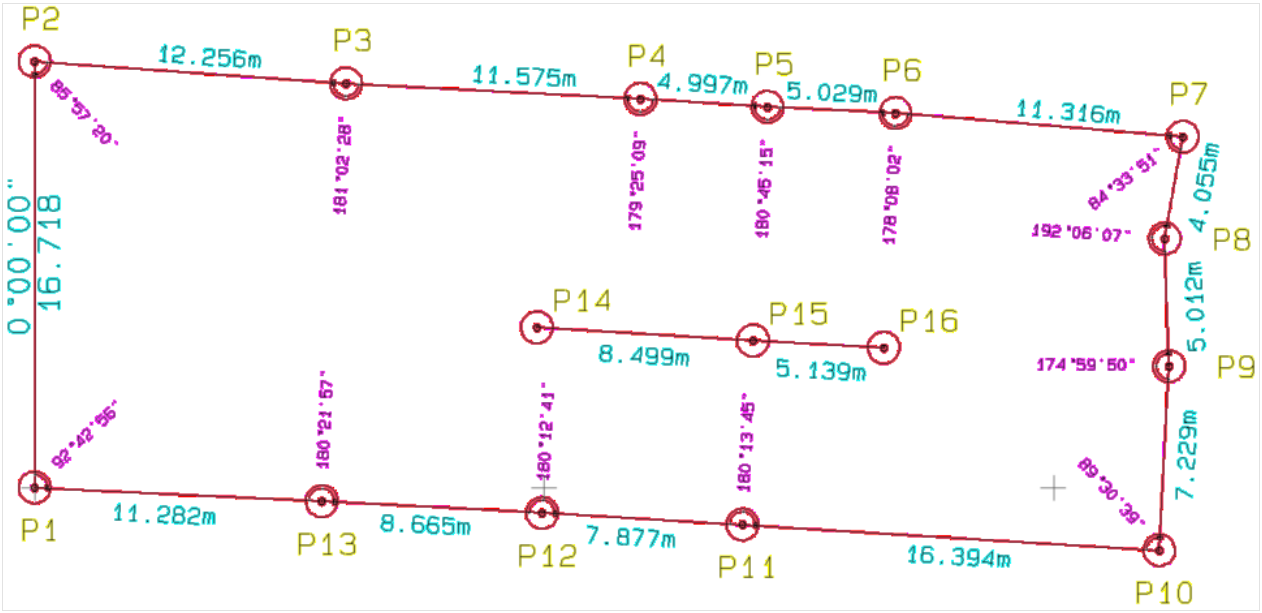


Figure 3.20 Projection du trajet désiré dans le plan du robot XOY

3.8.3 Test numéro 3

Le test 3 vise à réaliser un trajet en carré (trajet fermé) par le robot pour examiner la précision de ce dernier lors d'un retour au point de départ. Malgré le frottement, on a obtenu une bonne précision en passant à côté des points désirés (les points rouges) sur le trajet. Le robot est revenu au point de départ avec une précision d'environ de 60 cm. Cela est considéré comme une bonne précision, car la tolérance attribuée au plugin NavPointToPoint est de 1 m pour le rayon R autour des

points. On remarque dans le deuxième segment du trajet qu'il y un noeud sur le trajet, ce noeud est là pour réaliser un passage à proximité du point désiré. Le plugin NavPointToPoint a partiellement réussi à gérer le contrôle pour passer par les points désirés, car lorsque le robot suit une ligne droite, ce plugin émet une vitesse angulaire bien qu'il ne faille pas émettre ce type de vitesse. Cette dernière génère des erreurs accumulées dans la dynamique du système. Sur la figure 3.22, le trajet de l'odométrie est tout à fait loin de la réalité ; cela affecte le comportement du filtre et engendre un trajet déformé et divergent. Cependant, ce trajet ressemble par la forme au trajet réel. Comme dit auparavant dans la section 2.10 lors de l'expérimentation impliquant le capteur GPS, l'erreur du biais se répète pendant une période déterminée durant la journée ; cela apparaît clairement sur les figures 3.19 et 3.20, le biais est de ($x = 0.2m$,$y = 2m$).

Pour comprendre les résultats, il faut regarder les figures 3.19 et 3.20 afin de pouvoir comparer la carte géographique et avec les figures de la projection locale dans le plan XOY. Le positionnement des points GPS projetés montre que le biais soustrait du trajet GPS a donné de bons résultats lorsque le robot est passé par le côté gauche du quatrième segment sur les points $P11$,$P12$,$P13$. Pour les segments formés par les points $P2$,$P3$,$P4$, on remarque sur la figure 3.22 qu'aucun biais n'a été lié au signal GPS durant la navigation. Pour les points $P5$,$P6$,$P7$,$P8$,$P9$,$P10$,$P11$,$P12$ et $P13$, les données GPS présentent un biais qu'on enlève dès le début. Alors pendant la navigation sur ces quatre segments, on remarque que le biais soustrait au début représente un biais lié au signal GPS. Pour vérifier le passage par les points, une vidéo a été prise afin de valider l'algorithme de fusion. Quant au filtre de Kalman et la fusion de données, ce dernier a donné des résultats pertinents.

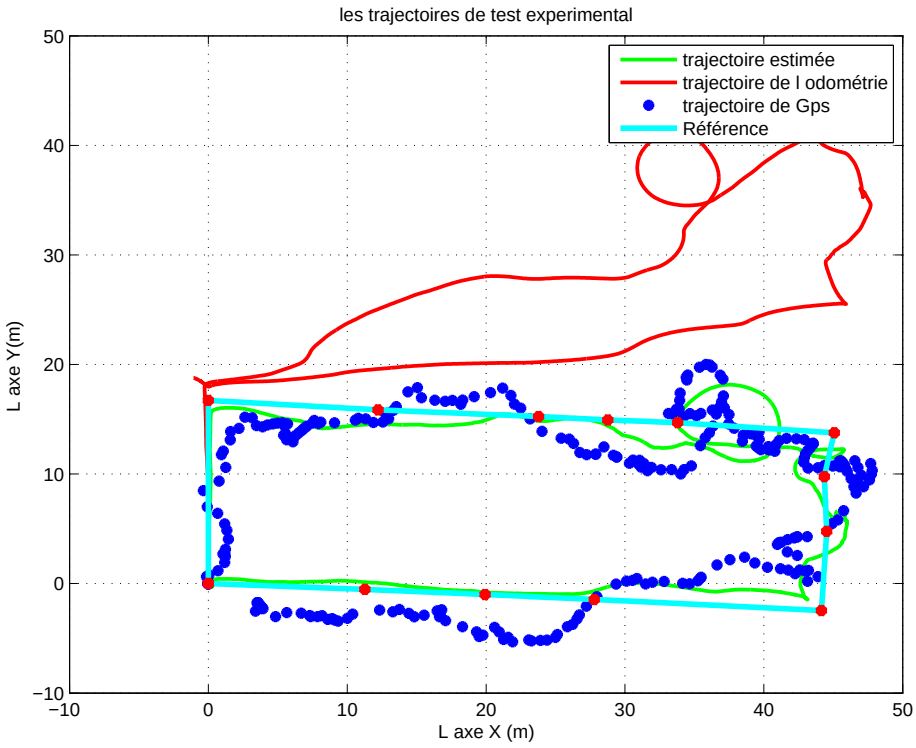


Figure 3.21 Trajet sous forme d'un carré

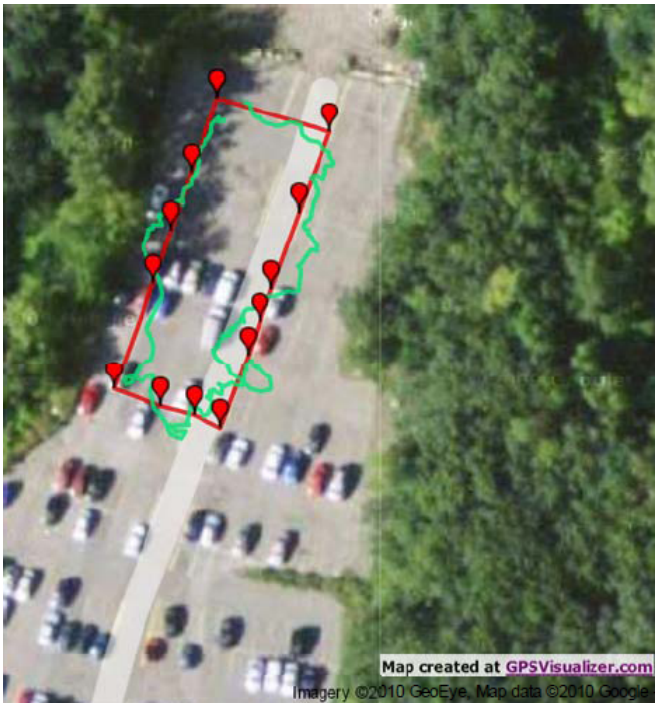
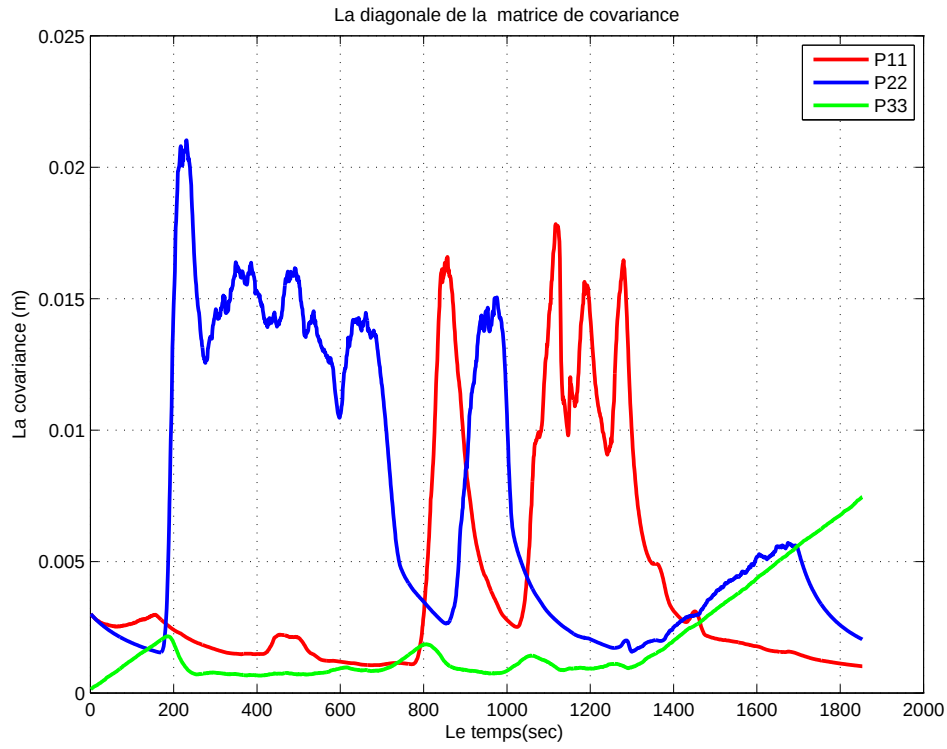


Figure 3.22 Trajet sur une carte géodésique

(a)



(b)

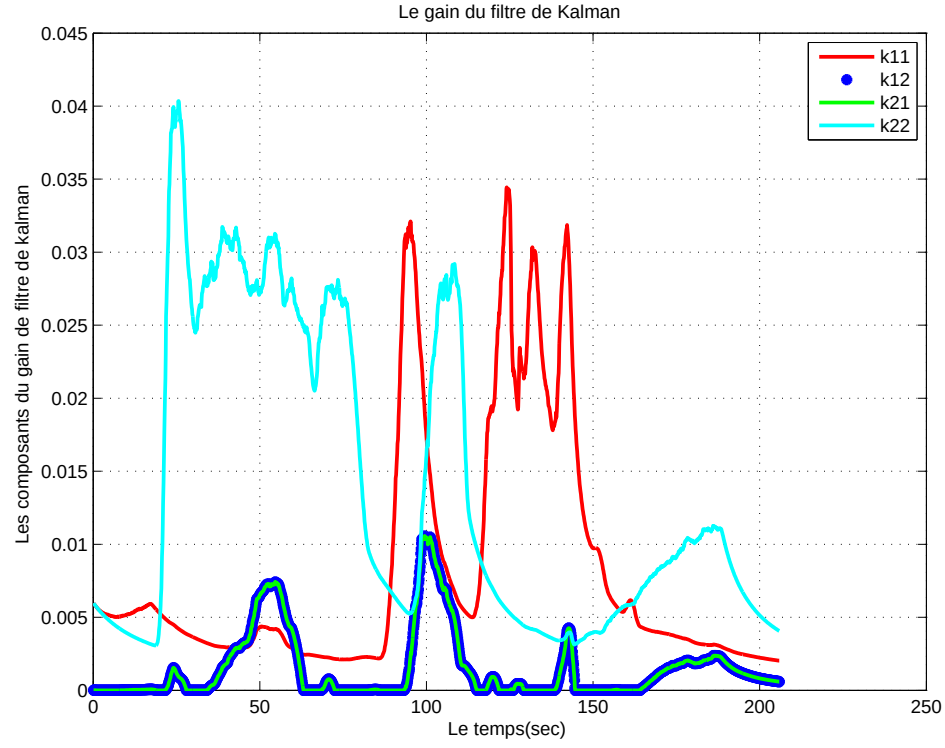


Figure 3.23 Diagonale de la matrice de covariance et le gain de filtre correspondant

3.8.4 Test numéro 4

Le but de ce test est de déterminer si le robot est capable de suivre une ligne droite ($45.14m$) comme ce serait le cas lors d'une navigation sur le trottoir en l'absence d'obstacles. Le robot démarre initialement avec une bonne précision, car l'estimé suit l'odométrie. Par la suite, le robot commence à s'éloigner du trajet désiré, cela revient au fait que l'estimé commence à suivre le capteur GPS. La tolérance du plugin NavPointToPoint est de 1.5 m pour le rayon des cercles entourant les points du trajet désiré. En général, la condition initiale du filtre joue un rôle important au niveau du comportement de ce dernier. La modification des paramètres initiaux de la matrice de covariance assure au filtre un comportement plus ou moins doux selon de la valeur de ces coefficients de départ. Par conséquent plus on effectue de tests et plus on ajustera au mieux ces paramètres initiaux. Le robot est arrivé au point final avec une bonne précision (50 cm), cela est d'autant plus satisfaisant qu'il s'agit en fait d'un retour vers le point de départ. La figure 3.24 montre les trois trajets et la projection du GPS dans le repère local du robot. Sur la figure 3.25, on remarque qu'il y un décalage de ($x = -0.1m$,$y = -2m$). Le décalage a été corrigé dès le départ, c'est pour cela que le robot est arrivé à la fin de trajet. En comparant les figures 3.24 et3.25, on constate que, sur la vidéo de ce test, la projection du GPS reflète la réalité notamment lorsque le robot arrive à la fin, pourtant sur la figure 3.25 le robot n'est pas arrivé au point indiqué dans la projection. La même chose s'applique pour le point de départ lorsque le robot revient à ce dernier. Le robot ne devrait pas dépasser pas ce point selon la projection des données GPS, pourtant sur la carte géodésique le robot le dépasse. On déduit de ce test que l'hypothèse attestant que l'erreur est approximativement fixe pendant une période déterminé de la journée est valide. Selon la figure 3.26, on a commencé le test avec une assez petite covariance initiale. Autrement dit, on fait plus confiance à l'odométrie qu'au capteur GPS. C'est pour cela que l'on a obtenu un estimé proche du trajet de l'odométrie au départ et que par la suite ce dernier a commencé à suivre le GPS. Dans un second test on va augmenter les valeurs initiales de la matrice de covariance pour donner la priorité au capteur GPS. Comme l'estimé suit le trajet désiré, la covariance diminue avec le temps. Le gain du filtre ressemble par la forme à la covariance (figure 3.27).

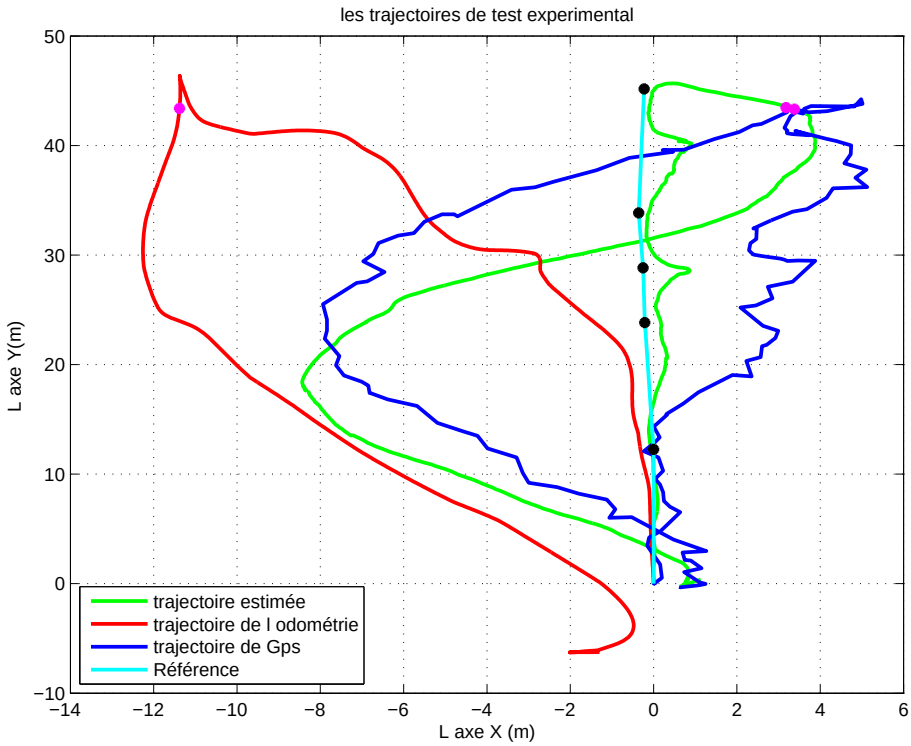


Figure 3.24 Trajet sous forme d'une ligne droite

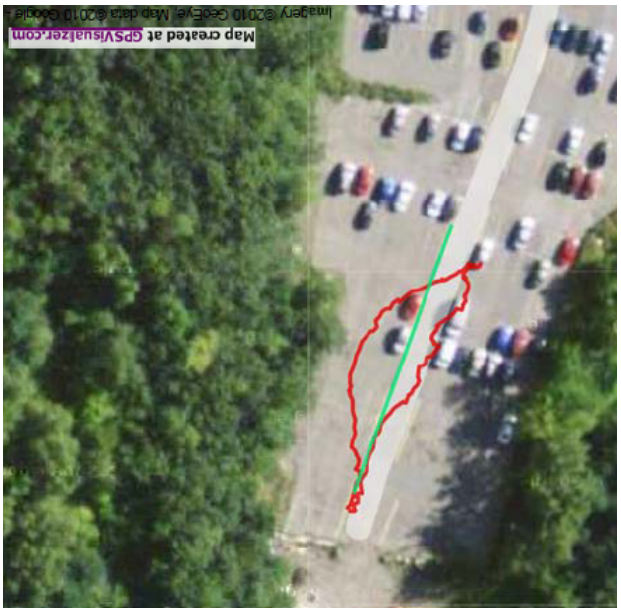


Figure 3.25 Trajet sur une carte géodésique

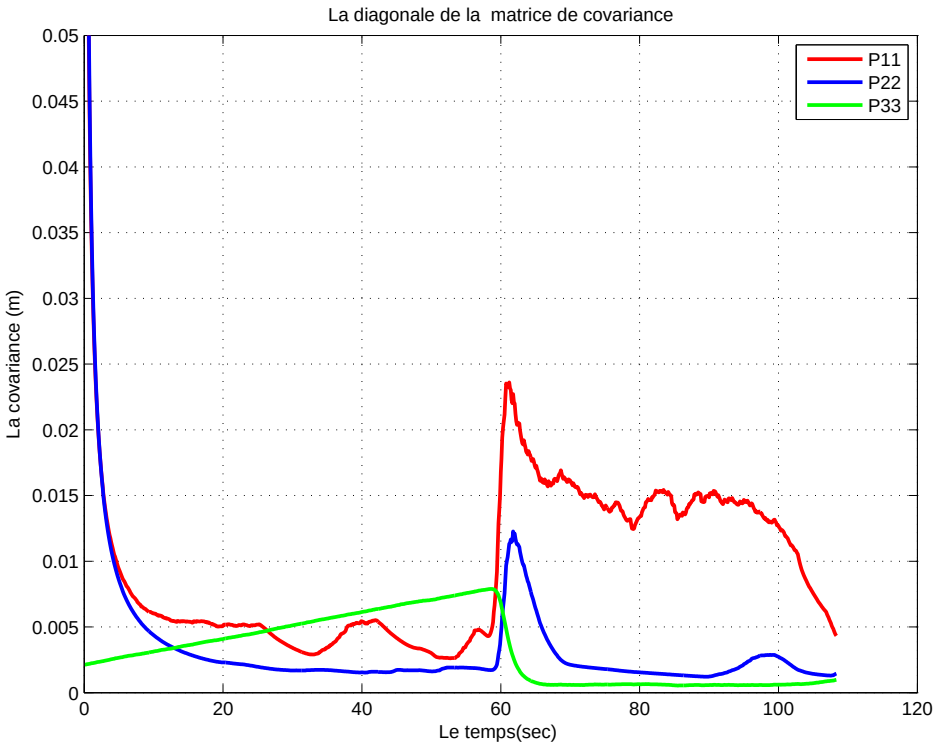


Figure 3.26 Éléments principaux de la matrice de la covariance

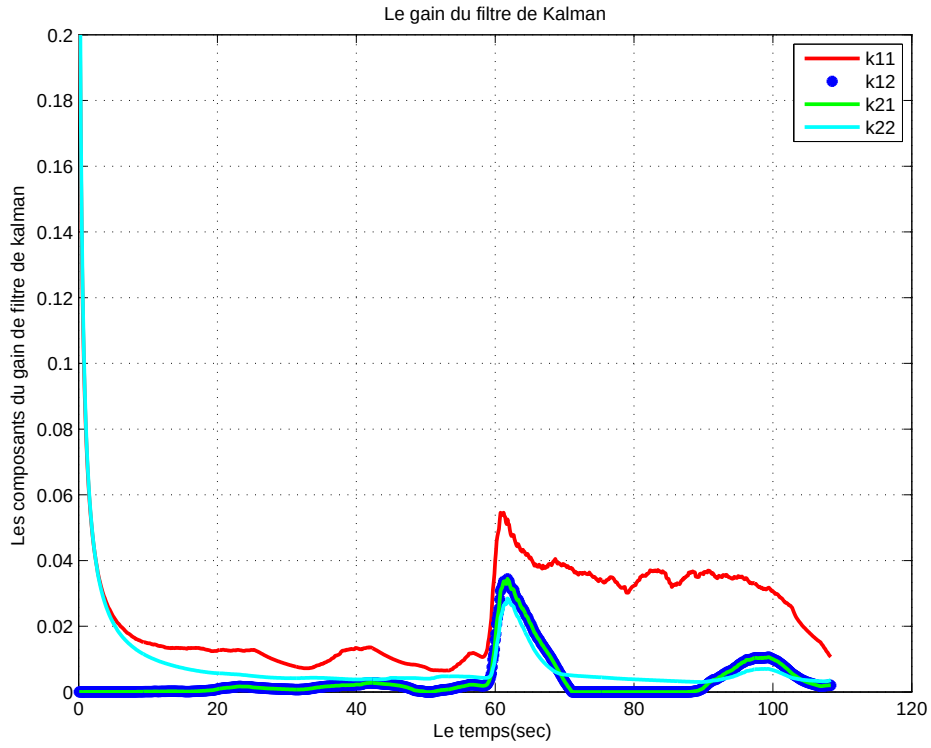


Figure 3.27 Éléments de la matrice du gain de filtre

La figure ci-après est la projection dans le repère local du robot, le point $P2$ est le point de départ.

Figure 3.28 Projection du trajet sous forme d'une ligne droite dans le repère local du robot

Pour améliorer les résultats, plusieurs autres tests ont été effectués par l'entremise de la variation des conditions initiales. Ainsi, on obtient de bons résultats puisque ce deuxième test moyennant un meilleur ajustement des paramètres initiaux améliore les performances du filtre. En effet, le robot a suivi la ligne droite jusqu'au bout. À la suite de ces tests complémentaires, on peut dire que le robot est capable de suivre le trajet désiré avec une bonne précision. La carte, sur la figure 3.30, montre une bonne concordance entre le trajet désiré et le trajet réel du robot. Le décalage dans ce test est de $(x = 1m, y = 0.3m)$. Le trajet GPS projeté dans le plan local représente la réalité vérifiée vidéo à l'appui. Avec le même argument, on peut vérifier les trajets de la figure 3.29. Le gain ressemble, par la forme, à la covariance mais avec une échelle différente (figures 3.31,3.32).

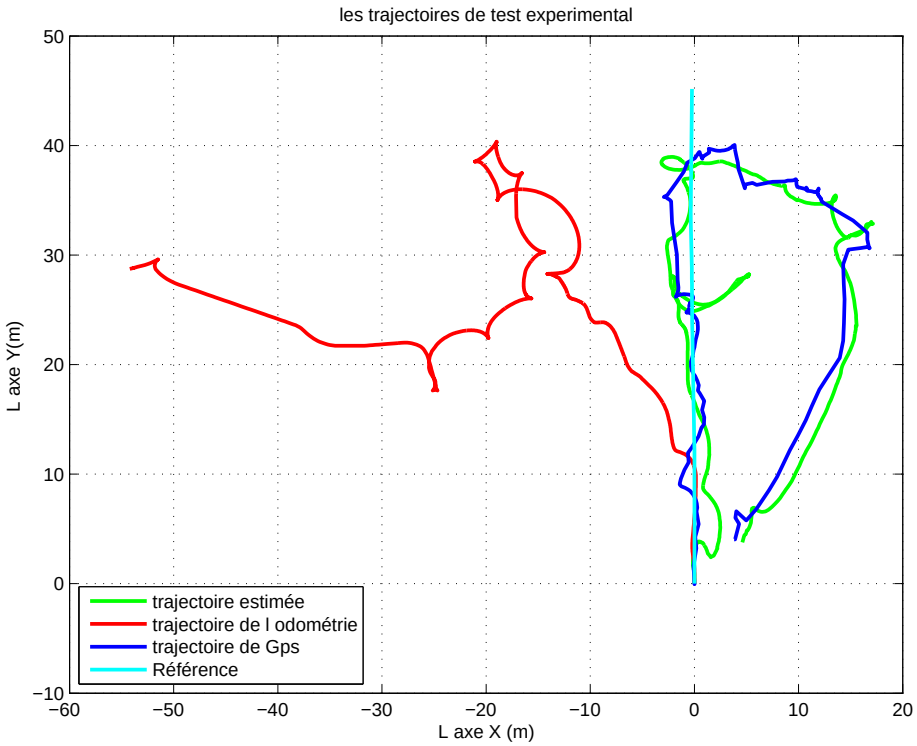


Figure 3.29 Trajet sous forme d'une ligne droite

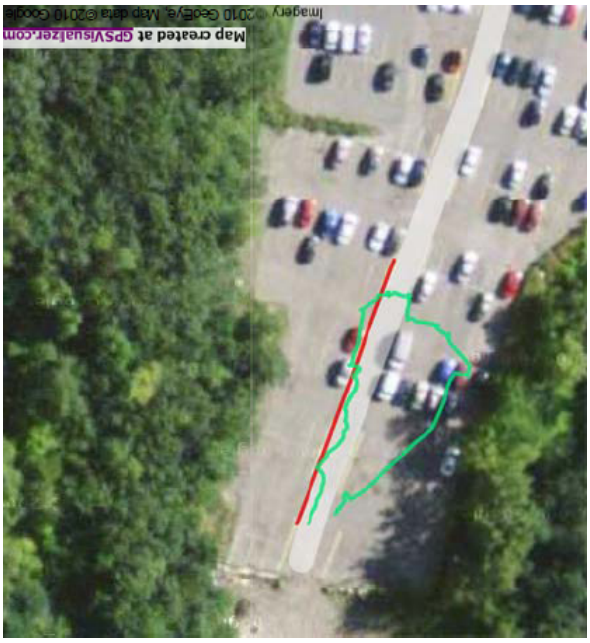


Figure 3.30 Trajet sur une carte géodésique

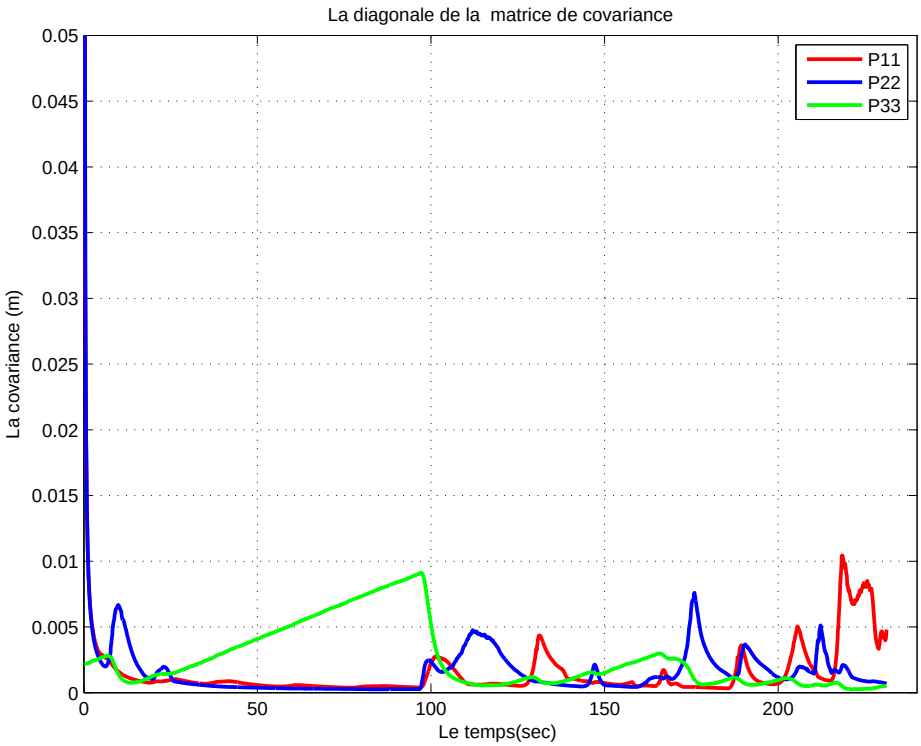


Figure 3.31 Éléments principaux de la matrice de la covariance

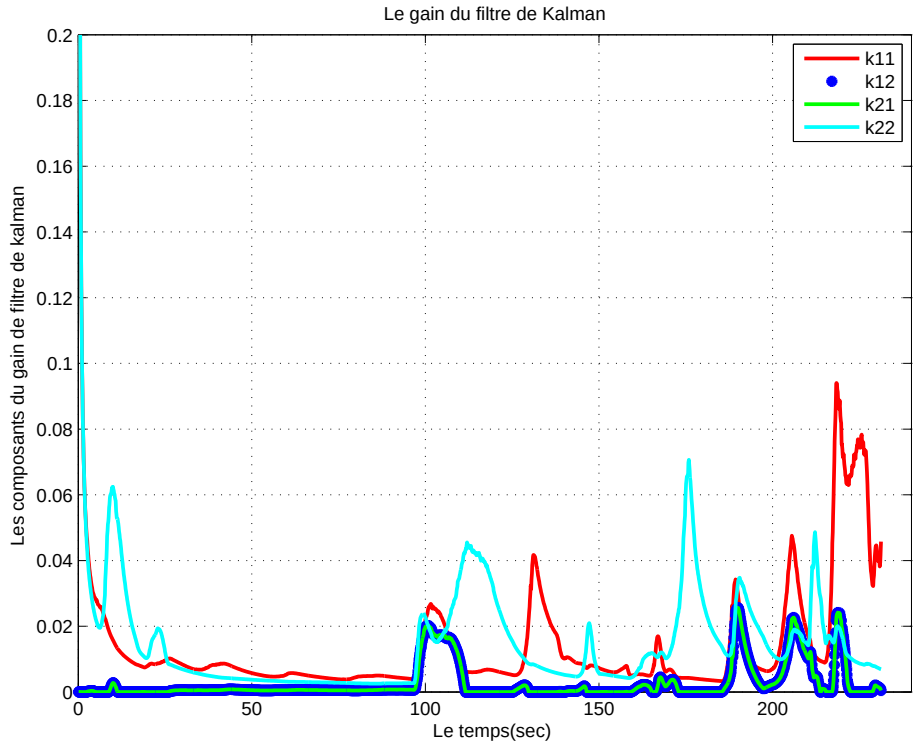


Figure 3.32 Éléments de la matrice du gain de filtre

3.8.5 Test numéro 5

Le test 5 représente le test final englobant toutes les différentes situations pouvant être rencontrées dans une navigation réelle (présence d'obstacles de dénivellation etc.). Il consiste en le suivi d'une trajectoire désirée (composé des points GPS) sur un trottoir (représenté par des plaques de polystyrène comme illustré sur la figure 3.33) en évitant les obstacles détectés. Dans la navigation en robotique, les obstacles négatifs ou positifs sont semblables du point de vu dimensionnel. C'est pour cela que l'on peut considérer les plaques de polystyrène comme des obstacles négatifs par souci de simplicité de représentation. Le suivi du mur est une partie de la tâche à effectuer pendant la navigation du trajet. Suivre un mur peut être vu d'une autre manière soit plutôt, celle de garder une distance fixe avec l'obstacle, en l'occurrence le mur dans notre cas, de sorte que le robot puisse naviguer avec une marge de distance. Néanmoins, si le robot commence à se rapprocher de l'obstacle, le mode réactif dans l'algorithme doit fonctionner et effectuer la manoeuvre nécessaire pour éloigner le robot le plus loin possible du danger représenté par une dénivellation. Le plugin NavPointToPoint détermine les vitesses linéaire et angulaire en fonction de ce point suivi. L'architecture de commande utilisée dans ce test sera expliquée plus en détail dans les sections suivantes. Au final, le robot a bien suivi la trajectoire désirée en détectant les obstacles et en les évitant. Les figures 3.34, 3.35 illustrent les trajets suivis géographiquement avec leurs projections dans Matlab. Les figures 3.37, 3.36 quant à elles montrent les covariances et les gains utilisés. Les gains $K11$, $K22$ ressemblent aux $P11$, $P22$ respectivement par la forme. À mentionner que, l'ajout d'une fonction représentant le frottement, dans l'équation qui calcule la covariance, pourrait améliorer l'allure du trajet du robot (le trajet de l'odométrie). De plus, suite au changement de précision du GPS et à cause du climat ainsi que de la présence des bâtiments (zone urbaine) ; le changement des valeurs de la matrice R peut jouer un rôle important dans le processus de positionnement du robot. La détection des obstacles, qui cache les satellites, peut être faite par une caméra ou d'autres appareils destinés à améliorer la navigation à l'extérieur.

Figure 3.33 Installations du test final à l'extérieur

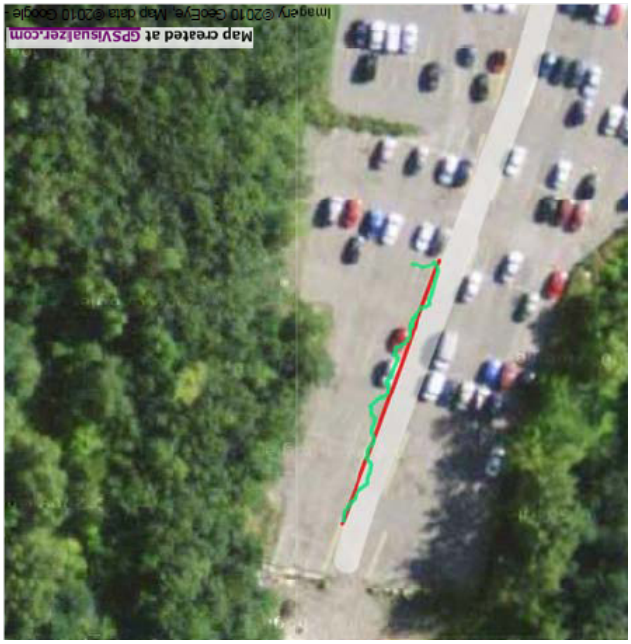


Figure 3.34 Trajet sur une carte géodésique

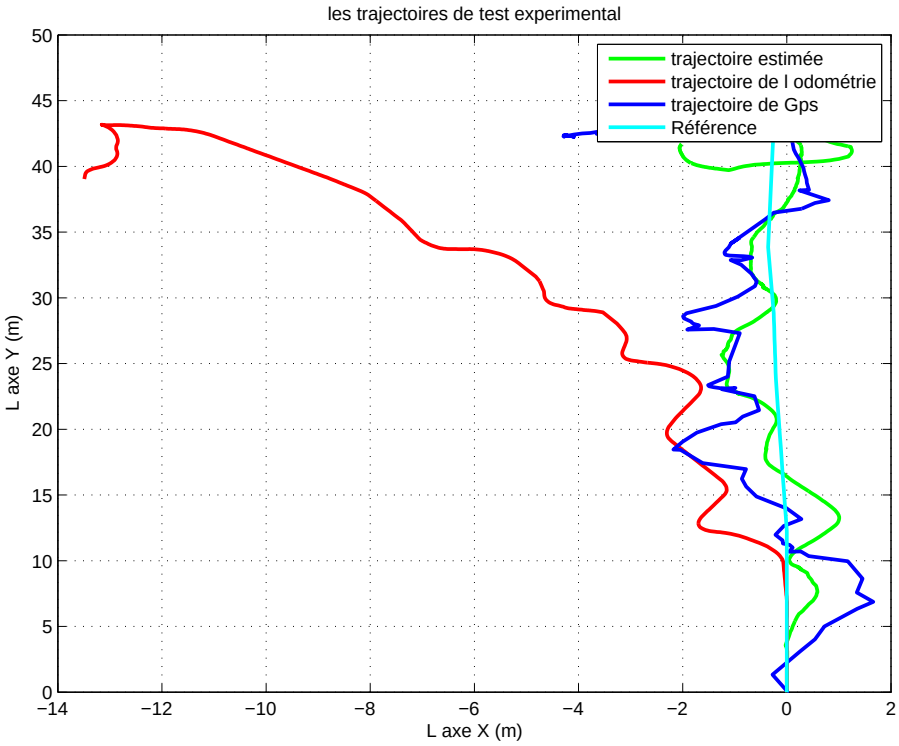


Figure 3.35 Trajet sous forme d'une ligne droite

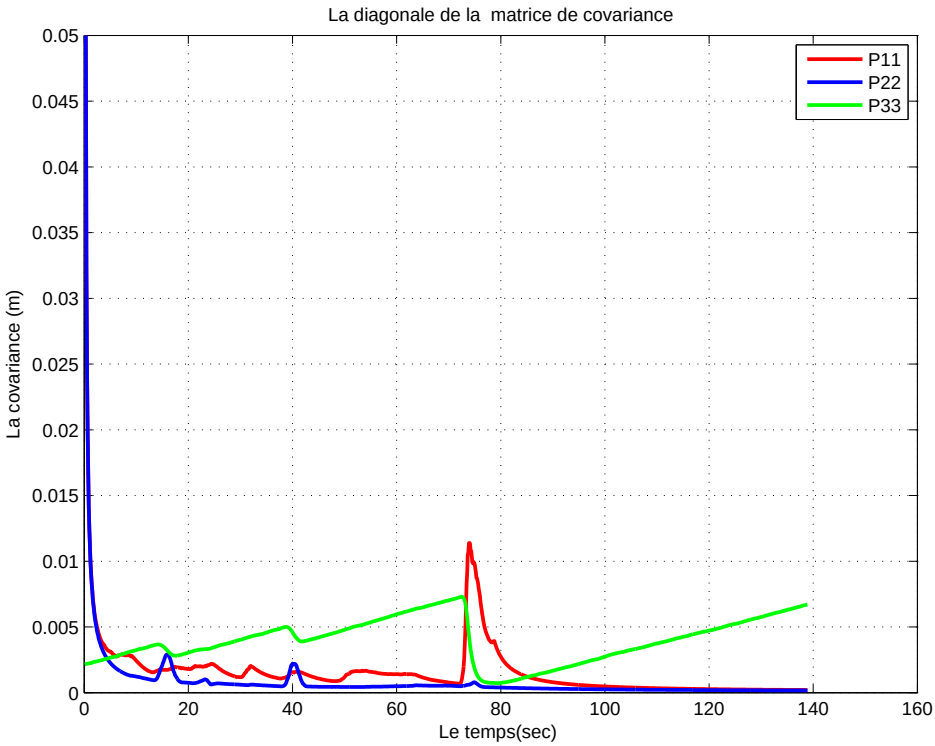


Figure 3.36 Éléments principaux de la matrice de la covariance

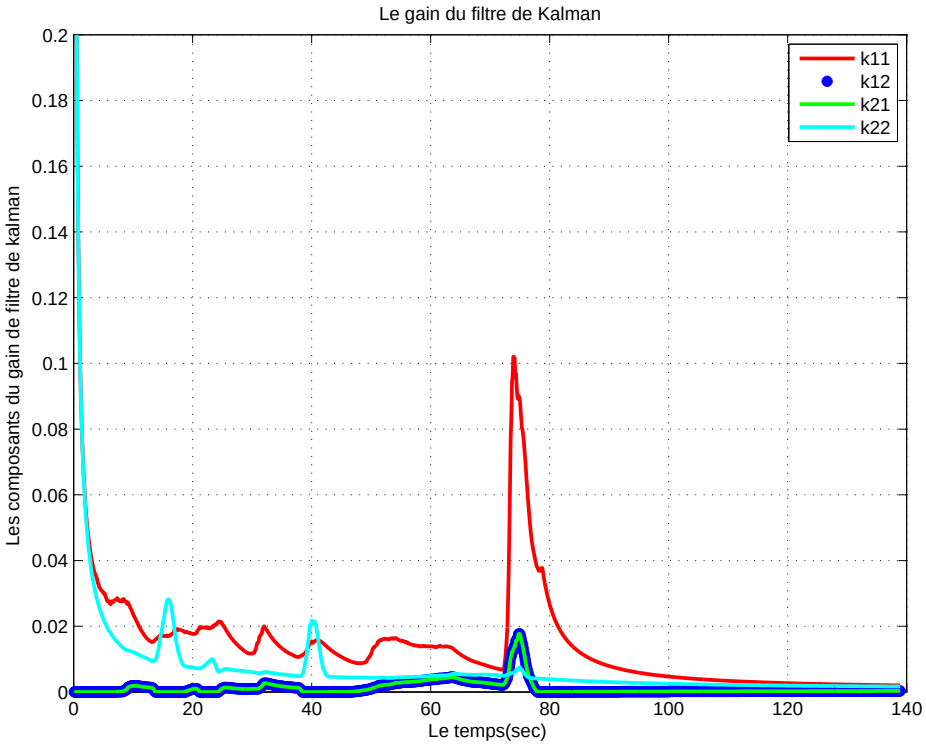


Figure 3.37 Éléments de la matrice du gain de filtre

3.8.6 Test numéro 6

Tel mentionné auparavant, une bonne initialisation et configuration du circuit Acropolis (voir annexe B pour l'architecture de commande) peut donner de meilleures performances. Par conséquent, la répétition des essais donne des valeurs de paramètres de plus en plus proches du réel. Ces paramètres représentent l'environnement dans lequel le robot progresse. Les résultats de ce dernier test sont assez bons et précis. La figure 3.38 dénote que le trajet parcouru correspond bien à la trajectoire désirée. En effet, Le fait de se localiser par rapport aux bords des trottoirs peut diminuer la déformation du trajet de l'odométrie. En conséquence de quoi, la fusion de données donnera de bons résultats. L'estimé calculé par le filtre est très proche du parcours désiré, cette affirmation étayée par la vidéo prise lors de ce test. Ce dernier illustre la capacité du robot à effectuer un trajet déterminé sur un trottoir réel. Le décalage étant de (x=0m,y=0.2m),sur la figure 3.39, une translation linéaire a alors été appliquée.

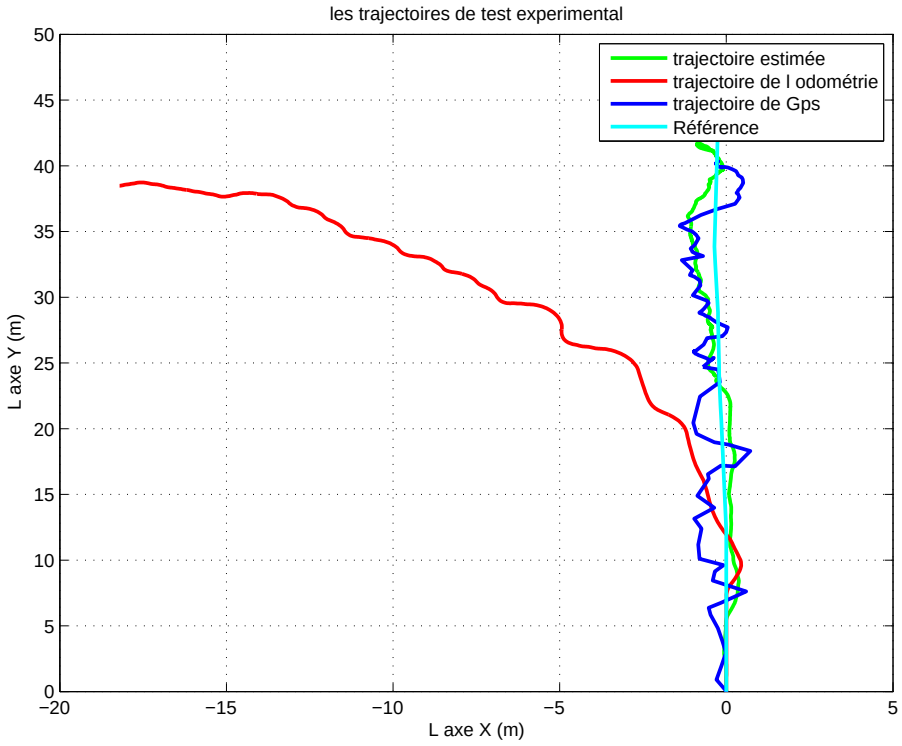


Figure 3.38 Trajet sous forme d'une ligne droite

Figure 3.39 Trajet sur une carte géodésique

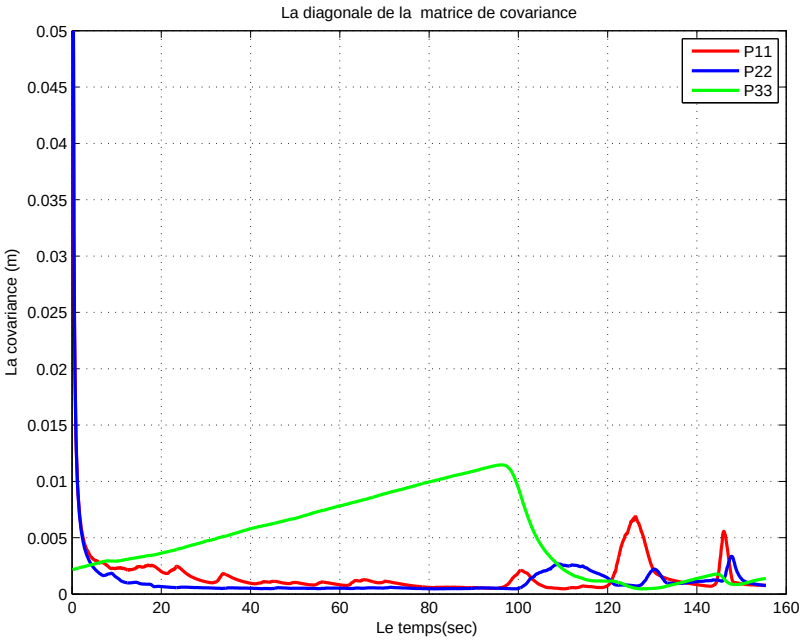


Figure 3.40 Éléments principaux de la matrice de la covariance

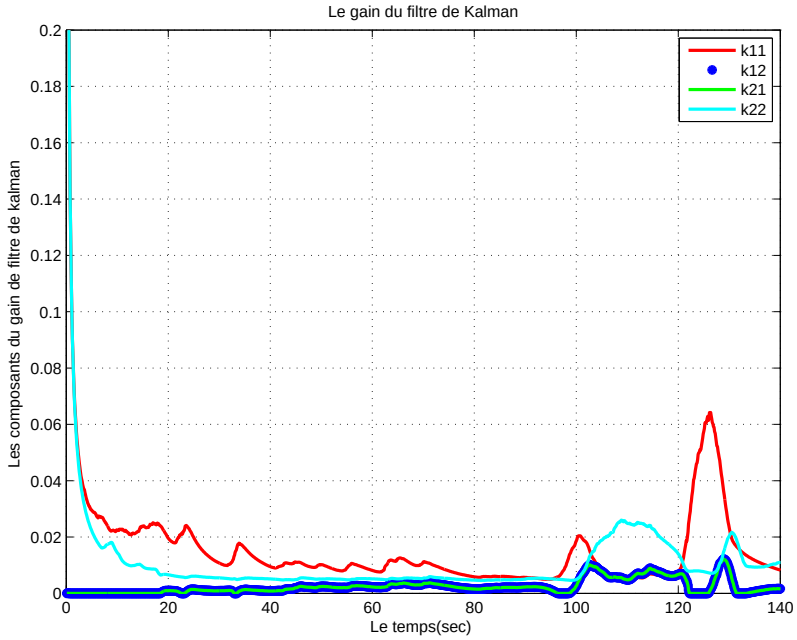


Figure 3.41 Éléments de la matrice du gain de filtre

3.9 Conclusion

Ce chapitre résume la fusion de données GPS et d'odométrie moyennant le filtre de Kalman étendu. Un développement des équations dynamiques du système a été présenté en détail afin d'avoir une idée de processus de la linéarisation et de la fusion pour combiner les deux sources de données. Le filtre de Kalman montre l'effet de navigation pour une longue distance sur le trajet estimé et la priorité attribuée à une source de données précise dans ce type de navigation. La simulation selon la méthode suggérée dans ce chapitre a été faite pour tester des algorithmes de navigation à l'extérieur. Les tests expérimentaux à l'extérieur sont faits dans des conditions différentes du climat ; les trajets désirés étaient un carré, une ligne droite et une ligne droite accompagnée par des plaques de polystyrène.

Le premier test a pour but de déterminer si l'algorithme en boucle fermée est capable de suivre un trajet désiré et revenir au point de départ avec une bonne précision. Le cinquième et sixième trajets sont faits pour simuler le trottoir. Le robot a été capable de suivre le trajet désiré avec succès. Ces tests peuvent être une bonne base pour faire la navigation dans des quartiers résidentiels avec une chaise roulante en évitant les obstacles. Cet évitement avec un seul laser incliné, afin de diminuer le coût, serait valide tant pour les obstacles positifs et que négatifs.

CHAPITRE 4

CONCLUSION

4.1 Synthèse des travaux

Dans ce travail, certains éléments des composantes d'un système de navigation autonome pour l'extérieur ont été développés :

- Les erreurs de GPS sont analysées afin d'en dégager les caractéristiques. Les tests réalisés à des moments différents et dans des conditions climatiques différents ont montrés que les erreurs combinent un biais variant lentement, et qui pourrait être compensé, et une composante aléatoire.
- Un moyen pour localiser et projeter ces données dans le repère local de robot connaissant la position initiale du robot a été présenté. Cette façon de faire implique une intervention de l'usager en début de parcours pour l'estimation de l'erreur initiale : c'est pour cela le repérage se fait par l'intermédiaire d'un navigateur GPS duquel ont extrait un fichier GPX (fichier de référence). Le fait d'utiliser les deux points de référence, dans le fichier GPX, rend le robot plus autonome en naviguant selon un trajet spécifié par l'usager. Après avoir acquis les données GPS et les avoir projetées dans le repère de robot ; une synthèse a été proposée pour compenser les erreurs dues aux biais provenant du signal GPS.
- La localisation finale du robot a été effectuée moyennant deux ensembles sensoriels : le premier est l'odométrie qui représente le modèle cinématique du robot ; le deuxième ensemble est le capteur GPS. L'odométrie accumule des erreurs lors de navigation sur de longues distances surtout dans un environnement non structuré. Pour remédier à ce problème, un capteur GPS commercial est ajouté pour tester l'efficacité de ce capteur à guider le robot dans son cheminement. La fusion de données par filtre de Kalman a montré son efficacité bien que le robot utilisé ait des erreurs de glissement significatives lors des manoeuvres.
- Pour réaliser les tests expérimentaux de façon autonome, le programme Acropolis, présenté à l'annexe B, est utilisé comme une plateforme pour
  - gérer l'acquisition des données à plusieurs niveaux dans l'architecture proposée,
  - traiter les différents types d'informations mesurées avec leurs bruits associés et
  - émettre les commandes correspondantes afin de réaliser la réaction requise permettant de suivre le trajet désiré tout en respectant les critères de sécurité requis.
- Un module pour suivre un mur a été mis en place pour compléter la tâche originale de ce projet. La détection d'obstacles a été développée de façon préliminaire pour des obstacles et

des dénivellations (voir annexe C).

4.2 Limitations de la solution proposée

Le biais associé au GPS peut être enlevé d'une partie de l'erreur mais cette solution n'est applicable sur le robot que pendant une certaine période au début de la navigation. Certains des tests expérimentaux ont montré que ce biais attaché peut faire diverger le robot de son trajet désiré. Le changement des conditions climatiques, le nombre de satellites disponibles, la période de navigation peuvent faire changer ce biais. C'est pour cela que le fait de changer le biais estimé au début de trajectoire rend le processus de positionnement plus efficace et plus précis mais ne règle pas le problème d'une dérive à plus long terme.

Bien que des tests impliquant le suivi d'un mur ont été réalisés avec succès, le problème du suivi d'un trajet dans un environnement urbain, même statique, avec de nombreux obstacles dont certains négatifs (trottoirs) reste entier. La navigation sur un trottoir est un défi complexe impliquant l'utilisation de plusieurs sources de données surtout dans ce type d'environnement non structuré.

4.3 Perspectives futures

Le projet de navigation à l'extérieur prend en considération que la détection de tout phénomène doit être traitée de façon urgente et l'algorithme développé doit réagir d'une façon fiable et rapide.

L'utilisation d'un seul appareil laser incliné pour détecter les obstacles positifs (objets, mur etc.) et négatifs (dénivellation, trou etc.) pourrait être une solution économique et intéressante. L'ajout d'une caméra pourrait aussi être envisagé : l'analyse d'images serait utile afin de d'apporter des informations complémentaires à celles obtenues avec le laser et détecter d'autres informations utiles : par exemple pour franchir une intersection ou traverser une rue.

Il pourrait être aussi utile d'utiliser une caméra pour détecter les dénivellations et fusionner ces données avec celles acquises par l'appareil laser incliné afin de construire une cartographie de l'environnement en question.

La détection des obstacles négatifs et positifs a été explorée à l'annexe C et des tests préliminaires ont été réalisés à l'intérieur dans le laboratoire. Il sera indispensable d'expérimenter plus à fond avant de se prononcer sur la validité de l'approche pour la navigation extérieure.

De plus pour augmenter la précision de l'estimation de filtre de Kalman ; l'utilisation d'un robot ayant moins de glissement lors des changements d'orientation contribuerait surement à une amélioration et pourrait permettent d'évaluer le plein potentiel de la fusion GPS/odométrie.

RÉFÉRENCES

ABDULLAH, M., STRANGEWAYS, H. et ZULKIFLI, S. (2010). Ionospheric differential error determination using ray tracing for a short baseline. *Advances in Space Research*, 46, 1326 – 1333.

BROWN, R., HWANG, P. et WILEY, J. (1992). *Introduction to Random Signals and Applied Kalman Filtering (2nd ed)*. Toronto : Wiley, c1992.

CHANG, T., HONG, T., LEGOWIK, S. et ABRAMS, M. (1999). Concealment and obstacle detection for autonomous driving. *Proc. of the Intl. Association of Science and Technology for Development-Robotics and Application*. Citeseer, 28–30.

COBOS, J., PACHECO, L., CUFI, X. et CABALLERO, D. (2010). Integrating visual odometry and dead-reckoning for robot localization and obstacle detection. *Automation Quality and Testing Robotics (AQTR), 2010 IEEE International Conference on*. IEEE, vol. 1, 1–6.

COMPANY GARMIN (2010). Gpx schema. Derived from :`http://www8.garmin.com/xmlschemas/GpxExtensions/v3/GpxExtensionsv3.xsd`. [Online ; accessed january septembre 2009].

DE SANTIS, R. M. (2006). *Commande des systèmes robotiques : cours ELE6207* . École polytechnique de Montréal. Département de génie électrique (2001).

DU PUY DE GOYNE, T. (2003). *Le G.P.S : Marine, Aviation, Randonnées*. Toulouse, France : Cépaduès-Éditions, c2003.

FEATHERSTONE, W. (1996). An updated explanation of the Geocentric Datum of Australia (GDA) and its effects upon future mapping. *Australian Surveyor*, 41, 121–130.

FOSTER, D. (2002). The gpx exchange format. Derived from :`http://www.topografix.com/gpx_resources.asp/`. [Online ; accessed october-2010].

GARMIN, C. (2010). Gps sensor. Derived from :`https://buy.garmin.com/shop/shop.do?pID=8631`. [Online ; accessed 30 septembre 2010].

JARVIS, R. (2010). Terrain-Aware Path Guided Mobile Robot Teleoperation in Virtual and Real Space. *2010 Third International Conference on Advances in Computer-Human Interactions*. IEEE, 56–65.

JUNG, B. et SUKHATME, G. (2004). Detecting moving objects using a single camera on a mobile robot in an outdoor environment. *International Conference on Intelligent Autonomous Systems*. Citeseer, 980–987.

KAPLAN, E. et HEGARTY, C. (2006). *Understanding GPS : Principles and Applications (2nd ed)*. Boston, Mass : Artech House, c2006.

KIM, S., ROH, C., KANG, S. et PARK, M. (2007). Outdoor navigation of a mobile robot using differential GPS and curb detection. *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 3414–3419.

LAUGIER, C., FRAICHARD, T. *ET AL.* (2001). Decisional architectures for motion autonomy. *Intelligent vehicle technologies : theory and applications*, 333.

LEE, S. et SONG, J. (2004). Mobile robot localization using optical flow sensors. *International Journal of Control, Automation, and Systems*, 2, 485–493.

LI, Q., YAO, M., YAO, X. et XU, B. (2010). A real-time 3D scanning system for pavement distortion inspection. *Measurement Science and Technology*, 21, 015702.

MAIER, D. et KLEINER, A. (2010). Improved GPS sensor model for mobile robots in urban terrain. *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 4385–4390.

OHNO, K., TSUBOUCHI, T., SHIGEMATSU, B. et YUTA, S. (2004). Differential GPS and odometry-based outdoor navigation of a mobile robot. *Advanced Robotics*, 18, 611–635.

PANZIERI, S., PASCUCCI, F. et ULIVI, G. (2002). An outdoor navigation system using GPS and inertial platform. *Mechatronics, IEEE/ASME Transactions on*, 7, 134–142.

PIÈPLU, J. (2006). *GPS et galileo :systèmes de navigation par satellites*. Paris : Eyrolles, c2006.

SCHNEIDER, A. (2002). Gps visualizer. Derived from :`http://www.gpsvisualizer.com/`. [Online ; accessed July-2009].

SCHON, S. (2006). Comparison of correction models for distance dependent systematic effects in GPS monitoring networks with large height differences. *Proceedings of the 3rd IAG Symposium on Geodesy for Geotechnical and Structural Engineering and the 12th FIG Symposium on Deformation Measurement*. PS19.1–PS19.9.

SINNOTT, R. (1984). Virtues of the Haversine. *Sky and telescope*, 68, 158.

SUZUKI, T., AMANO, Y. et HASHIZUME, T. (2010). 6-DOF Localization for a Mobile Robot Using Outdoor 3D Point Clouds. *Journal ref : Journal of Robotics and Mechatronics*, 22, 158–166.

TRUONG, N., AGASSOUNON, W., RESEARCH, A. T.-A. et MI., D. E. C. W. (2007). Utilizing Biomimetic Image Processing to Detect the Road Edge of Off-Road Terrain. Derived from :`http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA484745`. Online ; accessed august-2009.

VENESS, C. (2002). Calculate distance, bearing and more between latitude,longitude points. Derived from :http://www.movable-type.co.uk/scripts/latlong.html. Online ; accessed July-2009.

VINCENT, R., LIMKETKAI, B. et ERIKSEN, M. (2010). Comparison of indoor robot localization techniques in the absence of GPS. *Proceedings of SPIE*. vol. 7664, 76641Z.

WIJESOMA, W., KODAGODA, K., BALASURIYA, A. et TEOH, E. (2002). Laser and camera for road edge and mid-line detection. *Robot Motion and Control, 2001 Proceedings of the Second International Workshop Oct 2001*. IEEE, 269–274.

WIKIPEDIA (2010a). Bearing. Derived from :http://en.wikipedia.org/wiki/Bearing. [Online ; accessed july septembre 2008].

WIKIPEDIA (2010b). Global positioning system. Derived from :http://en.wikipedia.org/wiki/Global_Positioning_System. [Online ; accessed january 2010],Derived from :.

WILSON, M. et DICKSON, S. (1999). Poppet : A robust road boundary detection and tracking algorithm. *BMVC, TP Pridmore and D. Elliman, Eds. British Machine Vision Association*.

ANNEXE A

FILTRE DE KALMAN

Le filtre de Kalman (Brown *et al.*, 1992) est un outil mathématique conçu pour l'estimation de données bruitées avec des conditions initiales connue. En d'autres termes, le filtre de Kalman utilise les valeurs précédentes calculées de l'entrée pour évaluer les valeurs présentes de la sortie. Les deux propriétés importantes du filtre de Kalman sont :

– Un vecteur de modélisation de processus aléatoire sous conditions.
– Un processus récursif des données de l'entrée bruitées.

Ce qui suit est une introduction de notions essentielles de processus récursifs simples : En considérant le problème d'estimation de la moyenne d'une constante, aléatoire quelconque basé sur une séquence de mesures bruitées. Supposons que notre estimé soit la moyenne et que l'on veuille affiner cet estimé après chaque nouvelle mesure quand cette dernière est disponible. Les z_1, z_2, z_3, z_4,...,z_n sont les mesures prises à des instants discrets, une méthode pour traiter les données consiste à utiliser chaque mesure quand celle-ci est disponible pour calculer la moyenne correspondante avec l'algorithme suivant :

1. première mesure z_1 : mémoriser z_1 et estimer la moyenne comme

$$\widehat{m_1} = z_1$$

2. deuxième mesure z_2 : mémoriser z_2 en considérant z_1 et estimer la moyenne comme

$$\widehat{m_2} = (z_1 + z_2)/2$$

3. troisième mesure z_3 : mémoriser z_3 en considérant z_1, z_2 et estimer la moyenne comme

$$\widehat{m_3} = (z_1 + z_2 + z_3)/2$$

4. et cetera

Il est clair que la mémoire nécessaire pour sauvegarder les données sera grande avec le temps et aussi avec le nombre d'opérations arithmétiques requises. Pour palier à ce problème, on va appliquer un autre algorithme :

1. première mesure z_1 : calculer l'estimé comme $\widehat{m_1} = z_1$, mémoriser $\widehat{m_1}$ et négliger z_1.

2. deuxième mesure z_2 : calculer l'estimé comme une somme pondérée de l'estimé $\widehat{m_1}$ et la mesure courante z_2.
$$\widehat{m_2} = \frac{1}{2}\widehat{m_1} + \frac{1}{2}z_2$$
; mémoriser le $\widehat{m_2}$et négliger le z_2 et$\widehat{m_1}$.
3. troisième mesure z_3 : calculer l'estimé comme une somme pondérée de l'estimé $\widehat{m_2}$ et la mesure courante z_3.
$$\widehat{m_3} = \frac{2}{3}\widehat{m_2} + \frac{1}{3}z_3$$
; mémoriser le $\widehat{m_3}$et négliger le z_3 et$\widehat{m_2}$.
4. et cetera

Donc il est évident qu'à la *n*ième étape, la somme pondérée est :

$$\widehat{m_n} = \frac{n-1}{n}\widehat{m}_{(n-1)} + \frac{1}{n}z_n$$

Cette formule donne la même séquence de mesures sans être obligé de mémoriser toutes ces dernières. Le deuxième algorithme est un exemple simple d'un mode d'opération récursif. La clé de voûte dans n'importe quelle procédure est l'utilisation de l'itération précédente pour calculer les résultats désirés de l'étape courante. Afin d'appliquer la méthode récursive pour estimer un processus aléatoire, il faut préalablement que le processus et le bruit de mesure soient modélisés dans un vecteur.

Conception du filtre

Le processus aléatoire pour l'estimation peut être modélisé comme :

$$x_{(k+1)} = \phi_k x_k + w_k \tag{A.1}$$

L'observation (les mesures) du processus est supposée être produite aux points discrets à la même itération par une relation linéaire :

$$z_k = H_k x_k + v_k \tag{A.2}$$

où

- x_k=$(n*1)$ est le vecteur de processus à l'instantt_k.
- $\phi_k = (n*n)$, la matrice relie x_k et $x_(k+1)$en l'absence d'une fonction forcée.
  (si x_k est un échantillon de processus continu, ϕ_k est la matrice d'état de transition usuelle).
- w_k est $(n*1)$, le vecteur assumé une séquence blanche avec une covariance connue.

- $z_k = (m * 1)$, le vecteur mesuré à l'instant t_k.
- $H_k = (m * 1)$, la matrice donnant la connexion idéale (non bruité) entre les mesures et le vecteur d'état à l'instant t_k.
- $v_k = (m * 1)$, l'erreur des mesures, supposé être une séquence blanche avec une covariance connue et une corrélation zéro avec w_k.

Les matrices de covariance pour les vecteurs v_k et w_k sont donnée par

$$E(w_k w_i^T) = \left\{ \begin{array}{ll} Q_k & \text{if } i = k \\ 0 & \text{if } i \neq k \end{array} \right\} \tag{A.3}$$

$$E(v_k v_i^T) = \left\{ \begin{array}{ll} R_k & \text{if } i = k \\ 0 & \text{if } i \neq k \end{array} \right\} \tag{A.4}$$

$$E(w_k v_i^T) = 0 \tag{A.5}$$

pour tout k et i.

Supposant qu'on ait à un moment donné t_kl'estimé initial x_k^- du processus et que cet estimé soit basé sur la connaissance de ce dernier avant le t_k : le chapeau indique qu'il s'agit d'une estimation quant au signe moins, il indique que cet estimé est le meilleur du fait de l'assimilation des mesures à l'instant t_k. Posons la matrice P_k^-, d'erreur de covariance associée avec x_k^-. On définit l'erreur d'estimation :

$$e_k^- = x_k - x_k^- \tag{A.6}$$

Et la matrice de l'erreur associée :

$$P_k^- = E(e_k^- e_k^- T) = E((x_k - \widehat{x_k}^-)(x_k - \widehat{x_k}^-)^T) \tag{A.7}$$

Dans plusieurs cas, on commence l'estimation sans mesures préalables. Dans ce cas si la moyenne du processus est à zéro, l'estimé initial est aussi à zéro et la matrice d'erreur de covariance associée est la matrice de covariance de x elle-même. En supposant un estimé x_k^- à priori, on cherche à utiliser les mesures z_k pour améliorer l'estimé à priori. On choisit un mélange linéaire du bruit des mesures et de l'estimé à priori selon l'équation suivant :

$$x_k = x_k^- + K_k(z_k - H_k x_k^-) \tag{A.8}$$

Oú x_k est l'estimé actualisé, K_k le facteur de mélange (à déterminer ci-après) L'objectif maintenant est de déterminer le coefficient particulier $K_k$qui met à jour l'estimé et l'optimalise dans un certain sens. La matrice de covariance de l'erreur associée avec l'estimé mis à jour (posteriori) est :

$$P_k = E(e_k - e_k^T) = E((x_k - \widehat{x_k})(x_k - \widehat{x_k}^T) \tag{A.9}$$

En suite, on substitue l'équation A.2 à l'équation A.8,ensuite on substitue le résultat de l'expression x_k dans l'équation A.9.

$$P_k = E\left\{((x_k - \widehat{x_k}^-) - K_k(H_k(\widehat{x_k} + v_k - H_k\widehat{x_k}^-)) * ((x_k - \widehat{x_k}^-) - K_k(H_k(\widehat{x_k} + v_k - H_k\widehat{x_k}^-))^T\right\} \tag{A.10}$$

Maintenant, pour calculer l'espérance indiquée on remarque que$(x_k$ - $x_k^-)$, étant l'erreur d'estimation à priori, n'est pas corrélée avec l'erreur de mesures v_k. Ainsi l'équation A.10 devient :

$$P_k = (I - K_kH_k)P_k^-(I - K_kH_k)^T + K_kR_kK_k^T \tag{A.11}$$

On remarque que l'équation A.11 est une expression générale de la matrice de covariance de l'erreur mise à jour et elle est applicable pour chaque gain K_k qu'il soit optimal ou non. En revenant au problème d'optimisation, on désire trouver un gain K_k particulier qui minimise les termes individuels tout au long de la diagonale principale de P_k car ces termes représentent les variations de l'erreur de l'estimation pour les éléments du vecteur à estimer. L'optimisation peut être donnée par plusieurs méthodes. Il sera utilisé ici l'approche de calculs différentiels (straight-forward differential calculus approach). Pour cela nous aurons besoin de deux formules de différentiation matricielle

La première :

$$\frac{d[trace(AB)]}{dA} = B^T \tag{A.12}$$

(AB doit être carrée)

La deuxième :

$$\frac{d[trace(ACA^T)]}{dA} = 2AC \tag{A.13}$$

(C doit être symétrique)

Oú la dérivation d'un scalaire par rapport à une matrice est définie comme :

$$\begin{pmatrix} \frac{ds}{da11} & \frac{ds}{da12} & \cdots & \cdots \\ \frac{ds}{da21} & \frac{ds}{da22} & \cdots & \cdots \\ \frac{ds}{da31} & \frac{ds}{da32} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{pmatrix} \tag{A.14}$$

On réécrit maintenant l'équation A.11 sous la forme suivante :

$$P = P^- - KHP^- - P^-H^TK^T + K(HP^-H^T + R)K^T \tag{A.15}$$

On remarque que les deuxième et troisième termes sont linéaires en K et que le quatrième

terme est quadratique en K . On veut minimiser la trace de P car il représente la somme de l'erreur quadratique moyenne de l'estimé de tous les éléments de vecteur d'état. On peut utiliser l'argument énonçant que les erreurs quadratiques moyennes sont minimisées si le totale est minimisé, dès lors on aurait assez de liberté pour la variation de K . Par la suite, on continue le processus en différenciant la trace de$P$ par K , et en remarquant que la trace $P^{-}H^{T}K^{T}$ est égale à la trace de sa transposée KHP^{-} , on aboutit ainsi à l'équation suivante :

$$\frac{d[trace(P)]}{dK} = -2(HP^{-})^{T} + 2K(HP^{-}H^{T} + R) \tag{A.16}$$

On met la dérivative à zéro et on résout pour le gain optimale .Le résultat après substitution est :

$$K_k = P_k^{-}H_k^{T}(H_k^{T}P_k^{-}H_k^{T} + R_k)^{-}1 \tag{A.17}$$

Ce gain particulier minimise l'erreur quadratique moyenne est appelé Kalman Gain. La matrice de covariance associée avec l'estimé optimal peut être calculé. En revenant à l'équation A.11 on a :

$$P_k = (I - K_kH_k)P_k^{-}(I - K_kH_k)^{T} + K_kR_kK_k^{T} \tag{A.18}$$

$$P_k = P_k^{-} - K_kH_kP_k^{-} - P_k^{-}K_k^{T}H_k^{T} + K_k(H_kP_k^{-}H_k^{T} + R_k)K_k^{T} \tag{A.19}$$

En remplaçant l'équation A.17 à l'équation A.19 on a :

$$P_k = P_k^{-} - P_k^{-}H_k^{T}(H_kP_k^{-}H_k^{T} + R_k)^{-}1H_kP_k^{-} \tag{A.20}$$

oú

$$P_k = P_k^{-} - K_k(H_kP_k^{-}H_k^{T} + R_k)K_k \tag{A.21}$$

$$P_k = (I - K_kH_k)P_k^{-} \tag{A.22}$$

Parmi les trois expressions la troisième est la plus simple. Maintenant on a des moyennes pour assimiler les mesures à l'instant t_k. Cela se fait en utilisant d'une part $\widehat{x_k^{-}}$ puis P_k^{-} et d'autre part l'équation A.8 avec K_k étant le gain de Kalman tel donné par l'équation A.17 . Il est à noter qu'on aura aussi besoin de ces données à l'étape suivante afin d'utiliser les mesures $z_(k+1)$de manière optimale. L'estimé $\widehat{x_k}$ actualisé est aisément projeté par la matrice de transition. On a ignoré la contribution de w_k dans l'équation A.1 parce qu'elle a une moyenne nulle et qu'elle n'est pas corrélée avec aucun des w_i. Par conséquent on a

$$\widehat{x_k^{-}} = \phi_k\widehat{x_k} \tag{A.23}$$

La matrice de l'erreur de covariance associée avec $\widehat{x_{k+1}^-}$est obtenu par la formulation de l'expression de l'erreur priori

$$e_{k+1}^- = x_{k+1} - \widehat{x_{k+1}^-} = \phi_k \widehat{x_k} + w_k - \phi_k \widehat{x_k} = \widehat{x_k} e_k + w_k \tag{A.24}$$

On remarque maintenant que w_k et e_k ont zéro cross corrélation, car le w_k est le bruit de processus pour l'étape t_k, alors on peut écrire l'expression pour P_{k+1}^-

$$P_{k+1}^- = E[e_{k+1}^- e_{k+1}^T] = (\phi_k e_k + w_k)(\phi_k e_k + w_k)^T = \phi_k P_k \phi_k^T + Q_k \tag{A.25}$$

À présent, on a toute les quantités demandées à l'instant t_{k+1} et les mesures z_{k+1} peuvent donc être assimilées exactement à l'étape précédente. Alors les équations A.8,A.17,A.22,A.23,A.25 forment le filtre de Kalman Il est clair que dès que la boucle est mise en action, elle peut être continuée infiniment. Les équations pertinentes et les séquences des étapes de calcul sont clairement montrées dans la figure 3.1 . Ce qui conclut cette concise et synthétique présentation du filtre de Kalman

ANNEXE B

L'ARCHITECTURE DE LA COMMANDE

B.1 Introduction

Acropolis est l'environnement dans lequel est réalisée toute la partie expérimentale. Cet outil est conçu pour assurer des fonctions de control dans des projets robotiques de toutes sortes. Il est une alternative à l'environnement stage destiné à la simulation et complémentaire au programme player employé pour la configuration des appareils en robotique. Acropolis est installé sur les deux ordinateurs soit la plate-forme du robot et la machine de contrôle c'est-à-dire celle du développeur ; ceci en vue de permettre un contrôle partagé et une flexibilité lors des opérations de débogage ultérieures des différents algorithmes conçus.

B.2 Problématique

Les commandes émises au robot suivent plusieurs scenarios selon des critères imposés par l'usager. Par exemple, l'habilité de l'usager à conduire et à faire des manoeuvres difficiles qui impliquent une grande attention et des réactions actives au moment de la présence des obstacles mobiles ou à leurs approches. Une littérature importante et plusieurs recherches en cours existent pour optimiser la manière avec laquelle la commande doit être émise, c'est pour cela que nous n'allons aborder cet outil que de manière générale. Toutefois, notre approche s'articulera autour d'une structure bien posée donnant des résultats expérimentaux satisfaisants du point de vue de la détection des dénivellations et de l'évitement d'obstacles.

B.3 Architecture d'un circuit d'Acropolis

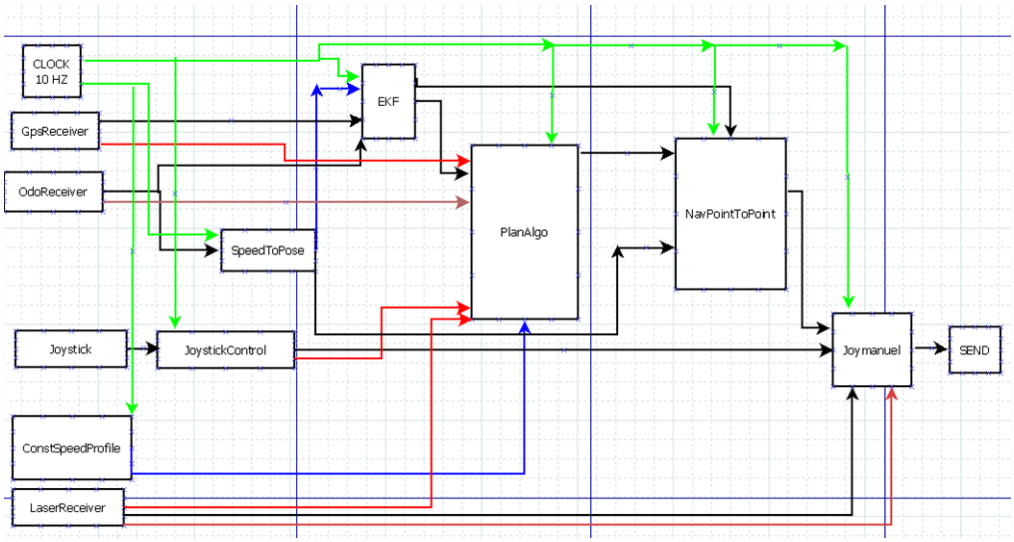


Figure B.1 Le circuit AcroPolis

Les plugins du circuit sont les suivants :

Clock : ce plugin a pour rôle de synchroniser les autres plugins entre eux, et d'activer la lecture des entrées des plugins à chaque cycle de balayage, l'Acropolis utilise une une fréquence de 10 Hz.

SpeedToPose : ce plugin change les vitesses linéaires et angulaires reçues du plugin OdoReceiver.

Joystick : ce plugin fait la lecture des données joystick pour les transmettre au plugin joystick control.

JoystickControl : fait l'interprétation des données nécessaires au plugin Joystick et issue d'une commande pouvant faire bouger le moteur

ConstSpeedProfile : donne un profil des vitesses désirées au plugin PlanAlgo.

LaserReceiver : fait l'acquisition des données laser pour les transmettre au plugin NavPointTo-Point et Joymanuel.

OdoReceiver : fait l'acquisition des données de l'odométrie pour le plugin EKF et SpeedToPose.

EKF : ce plugin représente le filtre de kalman.

PlanAlgo : ce plugin construit la commande et la transmet au plugin NavPointToPoint sous forme d'une liste de points.

NavPointToPoint : émet les commandes de vitesses linéaires et angulaires pour atteindre un point désiré.

Joymanuel : émet des commande direct au moteur par l'intermédiaire du plugin send.

Send : ce plugin donne une commande direct au moteur.

Le système fonctionne comme suit :

- Le plugin clock est connecté à tous les plugins pour la synchronisation de l'exécution des plugins.
- Le Gps Receiver donne la position de robot par le senseur Gps, le plugin EKF prend les vitesses linéaire et angulaire et fait l'estimation de la position ou bien la fusion de données entre les données de senseur Gps et la dynamique de robot qui représente l'odométrie.
- Le Plan Algo émet une MoveCommande en guise d'une liste des points de la trajectoire, dès qu'on a la commande émise par le NavPointToPoint, le plugin Joymanuel suit une politique spéciale pour émettre une commande propre selon le cas au moteur.

L'objectif de ce circuit est de développer une utilité robuste de navigation à l'aide d'un GPS non différentiel pour naviguer dehors par un robot mobile en détectant les dénivellations et en évitant les obstacles.

Le plugin PlanAlgo joue un rôle principal alors on va mettre l'accent sur ce plugin. Les tâches à faire par le PlanAlgo sont à réaliser en trois étapes :

1. La première étape est faite pour extraire les données (latitude, longitude) à partir des fichiers GPX. Ensuite, on utilise plusieurs lois pour calculer les distances entre les points GPS (latitude, longitude) et l'orientation entre les points. Ensuite on transforme ces distances en une liste des points (x, y) qui représente notre trajectoire. Le premier point dans la liste représente l'intersection entre les deux rues, donc on prend comme référence ce point.
2. La deuxième étape est d'estimer la position initiale de robot à partir des données fournies par l'usager par exemple la longue des deux rues qui font l'intersection et transformer les donnés de senseur GPS dans un repère local (repère du robot).
3. La troisième étape est pour générer la liste des points à naviguer (MoveCommande) en prenant en considération les donnés Laser et l'odométrie pour régénérer la liste et faire un évitement d'obstacle non réactive (d'avance). De plus, le PlanAlgo met à jour la liste des points dans le cas d'une commande générée par le joystick.

B.4 Architecture de la commande

Dans l'environnement Acropolis, certains types de commande sont à organiser pour contrôler le robot. La commande est générée à ce niveau dans le plugin de sortie Joymanuel comme suit. Le programme commence par faire un scan pour détecter la première dénivellation ($Den1$: à droite) en cas d'échec on peut alors déduire qu'il n'y pas de dénivellation à droite. Par conséquent, on

attribue au paramètre *Den*1 une valeur initiale aléatoire autour de 65°degré (cette valeur d'angle a été déterminée expérimentalement) suffisante pour garantir une navigation sécuritaire de ce côté. Dans le cas où il y une dénivellation à droite *Den*1, on procède à la seconde étape pour détecter la deuxième dénivellation *Den*2 et cela de la même façon que précédemment. Dans la possibilité où il y n'a pas détection d'une autre dénivellation *Den*2, on attribue alors à ce paramètre une valeur soit autour de 120 degré, puis on calcule les *MRDG*, *MRDD*. Si la *MRDG*(*MRDD*) n'est pas dans la limite autorisée, il faut donner une décision pour aller à droite (à gauche) D'une autre coté si la *MRDG* (*MRDD*) est dans les limites permises alors le robot peut continue tout droit. Quand il y a détection des dénivellations, les commandes s'exécutent directement. Le contrôle revient à l'usager qui utilise le Joystick et donne les commande directement au moteur ceci quand il n'y pas d'obstacles à proximité. Dans la possibilité d'une collision imminente avec un obstacle proche lors d'une navigation autonome la procédure change pour se laisser remplacer par une manoeuvre réactive garantissant un franchissement et évitement sécuritaire Si l'obstacle détecté se trouve à la gauche du robot, la commande émise doit être vers la droite et vice versa. Dans le cas où une commande de Joystick est émise par l'usager, autrement dit lors de l'activation du mode manuel ; l'algorithme doit vérifier qu'il n'existe pas d'obstacles à proximité qui peuvent engendrer une collision. En réaction à une telle situation, à savoir la détection d'un obstacle proche ; le contrôle sera transféré du Joystick vers les commandes directes impliquant les périphériques de navigation semi-autonome pour accomplir cette maoeuvre d'évitement. Une fois l'obstacle contourné, si l'usager sollicite toujours le mode manuel, le contrôle repasse au Joystick. Cette combinaison d'architecture de commande est faite pour conjuguer et associer plusieurs niveaux de commandes dites collaboratives. La commande collaborative repose sur une hiérarchisation des actions a entreprendre basée sur des priorités définies à priori par la pratique.

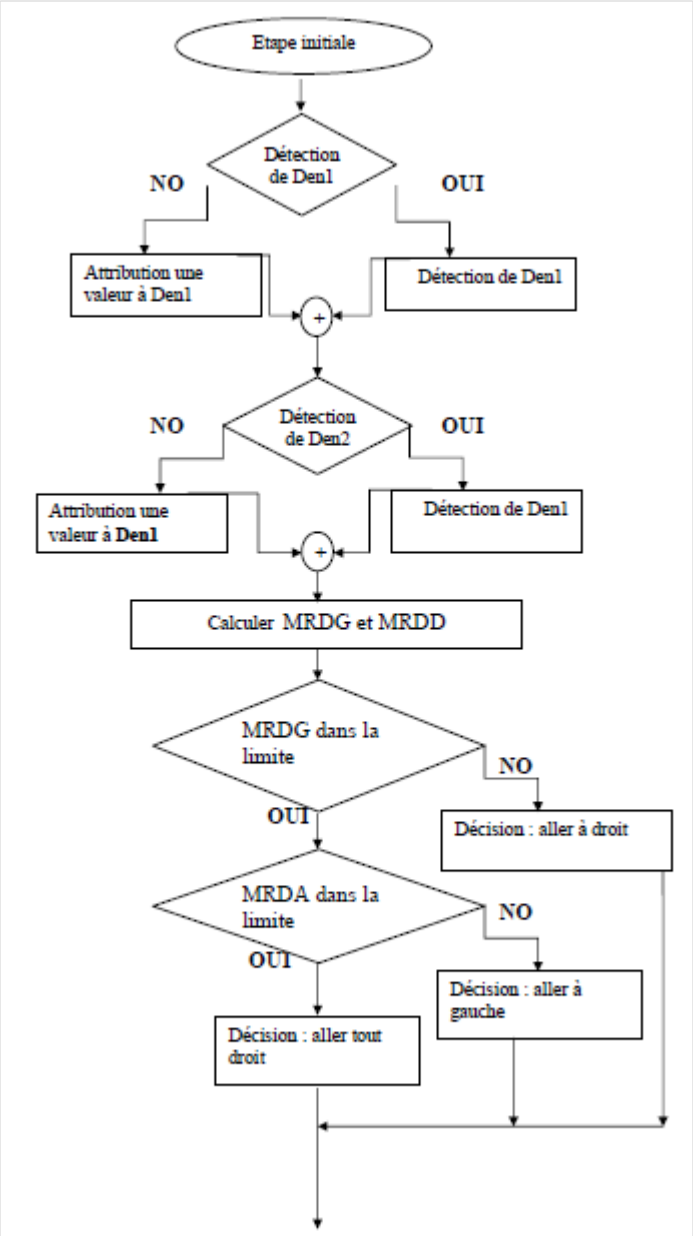


Figure B.2 Diagramme fonctionnel pour l'architecture de la commande

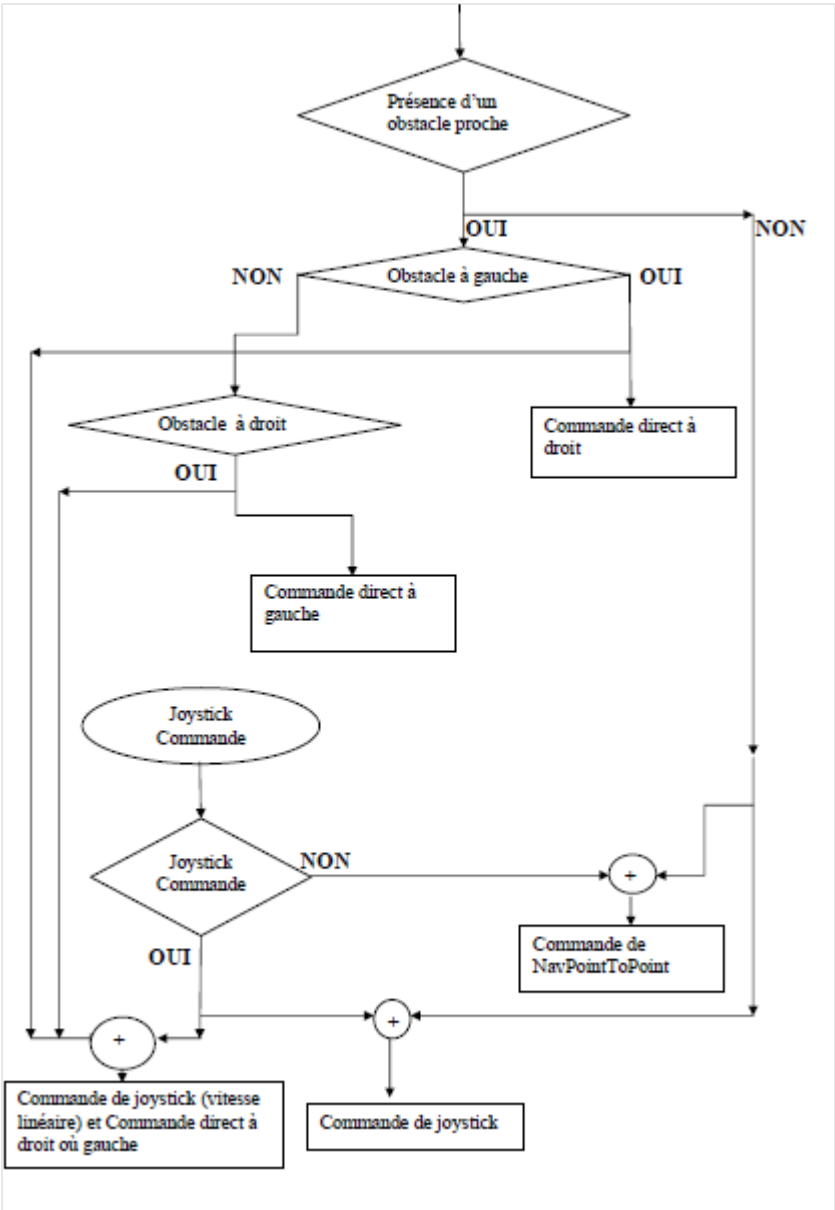


Figure B.3 Un diagramme fonctionnel pour l'architecture de la commande

B.5 Avantages de cette structure

Selon la structure représentée dans le diagramme fonctionnel des figures B.2 et B.3 , la commande est faite comme mentionné dans le plugin de Joymanuel (le plugin de sortie : supposé donner une commande vers le moteur). Dans l'explication dédiée à ce plugin nous avions commencé par faire un test pour détecter les dénivellations *Den*1 et *Den*2, ensuite pris une décision pour apporter la correction dans le sens correspondant. Il y a une différence entre la décision et la commande.

La décision d'aller à droite, à gauche ou bien de continuer tout droit est sujette à un changement dans le cas de l'apparition d'une décision plus prioritaire provenant par exemple de la détection d'un obstacle plus proche que la dénivellation. Les obstacles proches sont jugés plus importants à considérer pendant le processus de navigation. Donc s'il y des obstacles proches, une commande correspondante sera envoyée au moteur pour éviter une collision. Dans le cas d'absence d'obstacles, la commande est tout simplement relayée au plugin NavPointTopoint qui émet une consigne correspondant à la trajectoire calculée à partir des points de la liste. La commande de Joystick est sujette à une étude très longue car il s'agit d'une commande émise par l'utilisateur. Cette dernière doit toutefois respecter certains critères de sécurité par exemple etc. Cette partie du projet dépasse les le cadre du présent travail, elle ne sera donc pas abordé dans la suite de ce rapport. La stratégie suggérée est d'utiliser un principe dit de priorité dans le processus de navigation qui accorde la priorité la plus élevé aux réactions induites par les obstacles les plus proches. En effet, l'usager serait en mesure de diriger le robot où il voudrait sauf si un obstacle proche survenait soudainement ou bien si une dénivellation apparaissait pendant que le Joystick émet une commande. Dans ce cas le contrôle sera remis à une petite boucle sécuritaire qui transmettra des commandes directes au moteur pour effectuer un évitement rapide et sécuritaire. Par la suite si l'évitement est fait et que la commande du Joystick est encore présente, le contrôle revient à l'usager. Cette architecture d'Acropolis a divisé la commande en plusieurs niveaux selon un protocole de priorité capable de rencontrer les spécifications désirées lors du processus de navigation. Cependant le programme Acropolis a plusieurs défauts ,par exemple ,le cycle de scan d' Acropolis varie selon la largeur de programme et selon aussi la larguer du circuit ,Cela est d'autant plus vrai que les autres plugins sont plus grands que le plugin GpsData et c'est pour cela qu'une librairie spécifique et certains traitements particuliers sont nécessaires pour extraire les données GPS.

B.6 Applications de l'algorithme de détection des obstacles

Cet algorithme peut être utilisé pour la navigation à l'intérieur soit pour l'évitement des obstacles où bien pour faire un passage par la porte avec un laser Sick incliné surtout que cette dernière est une tâche difficile à réaliser. Comme mentionné plus haut, la détection des deux dénivellations est faite par constatation de variations de distances. Le robot évolue alors sur un couloir délimité par des marges de distance par rapport aux bords du trottoir. La procédure est la même pour le cas d'une porte à franchir. Ainsi, le robot calcule la distance par rapport au bord de la porte et passe par cette dernière avec une vitesse linéaire maximale. Si le robot n'est pas aligné avec la porte, ce dernier doit faire une manoeuvre et cela avant d'arriver à cette porte. Alors, il tourne et se réoriente pour s'aligner, ce qui garantit un passage rapide et sécuritaire.

B.7 Conclusion

Dans cette partie, un circuit a été implanté dans le programme de contrôle AcroPolis afin de commander le robot en boucle fermée. AcroPolis est un environnement multitâche pour lire, traiter les données et émettre les commandes. Une tâche a été assignée à chacun de ces plugins selon la fonction de ce dernier dans cette architecture. L'architecture d'AcroPolis est bien divisée dans le temps pour qu'il ecxécute les missions requises. Le plugin qui émet les commandes prend en considération la priorité du processus à faire afin de ne pas avoir de commandes contradictoires surtout lors du traitement de la détection des dénivellations et les décisions prises par les plugins précédents.

ANNEXE C

DÉTECTION ET D'ÉVITEMENT DES OBSTACLES

C.1 Introduction

Dans la revue de littérature, on a présenté des travaux dont le sujet est la détection des obstacles par un laser incliné. Dans cette revu, l'étude concentre sur la détection des dénivellations toute seule ; pourtant il n'y pas de définition ou un algorithme pour combiner les deux tâchés ensemble.

Ces travaux ne sont pas nombreux, c'est pour cela on présente dans ce chapitre l'évitement des obstacles dans le cadre d'utiliser un seul laser pour faire deux objectifs. L'un est de détecter les obstacles positifs ; l'autre est de détecter les dénivellations ou les trous qui représentent un danger.

Le fait d'installer trois laser sur une chaise roulante est une solution très couteuse et même n'est pas pratique du point de vu commercial. Le laser incliné peut réaliser le travail demandé avec sucées mais aussi il y des limitations prises en considérations lié avec l'environnement.

C.2 Définition et principes

C.2.1 Évitement d'un obstacle

Un des défis importants à relever en navigation robotique est la détection d'obstacles Pour cela, il faut définir l'obstacle et mettre en place un algorithme pour l'éviter.

C.2.2 Définition d'un obstacle

L'obstacle est un objet qui empêche le mouvement sur une trajectoire planifiée et qui cause dans certains cas un accident variant en dangerosité selon l'environnement où le robot progresse. La définition d'un obstacle varie selon l'appareil qui détecte ce dernier. Par exemple, la manière dont on définit un obstacle dans un algorithme avec un appareil laser orienté horizontalement est différente de celle d'un laser incliné. La méthode suggérée ici est d'utiliser un laser incliné, cela implique que les obstacles sont définis selon deux catégories :

- obstacles négatifs ;
- obstacles positifs.

C.2.3 Obstacles négatifs

Les obstacles négatifs sont détectés par un laser incliné d'un certain angle. Ces derniers sont, du point de vue du laser, interprétés comme étant des variations positives entre deux rayons voisins dans une région scannée, tel qu'illustré à la figure C.1 :

$R2 > R$ et $R < R1 \Rightarrow$ obstacle négatif dans une région scannée et délimitée par l'algorithme.

où R est le rayon de référence, autrement dit le rayon du laser qui touche la surface du plancher.

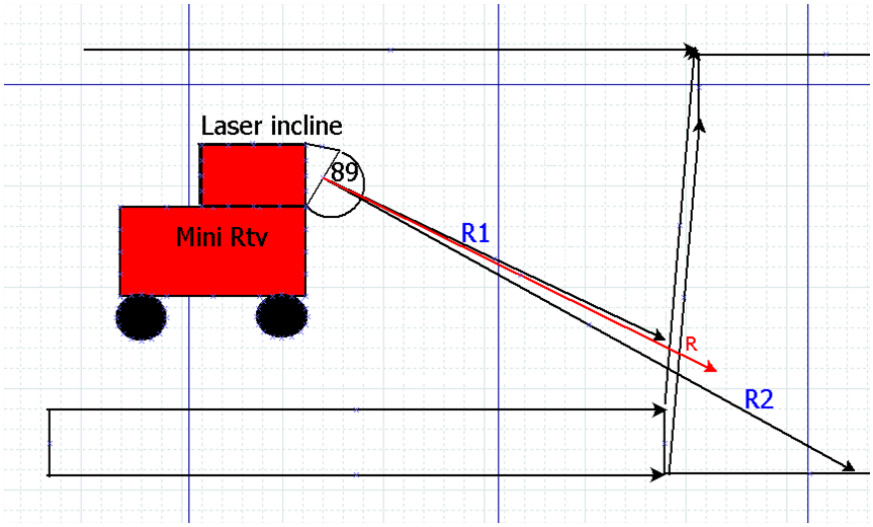


Figure C.1 Détection des obstacles négatifs

C.2.4 Obstacles positifs

Les obstacles positifs, détectés par un laser incliné, dont le plan x, y est perpendiculaire à l'axe z du repère lié au capteur sont, du point de vue du laser, quant à eux, interprétés comme étant des variations négatives entre deux segments voisins dans une région scannée, tel qu'illustré à la figure C.2 :

$R < R1$ et $R > R2 \Rightarrow$ obstacle positif dans une région scannée et délimitée par l'algorithme.

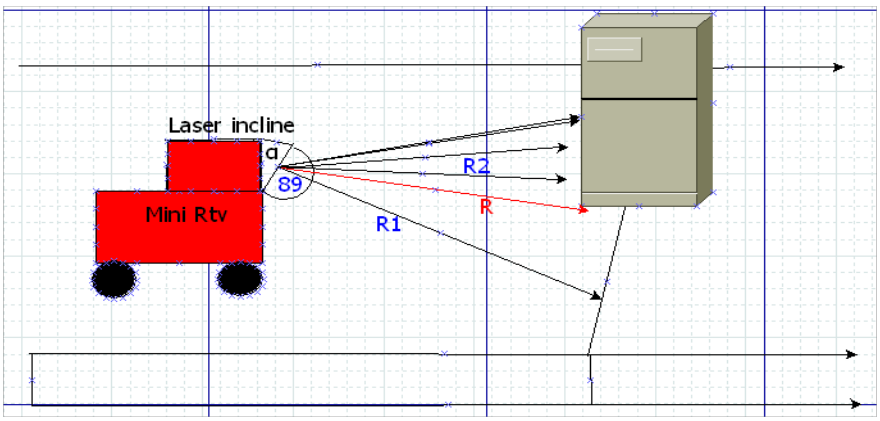


Figure C.2 Détection des obstacles positifs

C.2.5 Détection des obstacles négatifs avec un laser incliné

L'objet de notre travail est la détection des obstacles négatifs en vue de faire progresser le robot de la manière la plus sure qui soit et cela dans un environnement restreint, dangereux et plein d'obstacles imprévus. La procédure de détection de ces obstacles est basée sur l'utilisation d'un laser incliné d'un angle α détectant les objets qui se situent devant le robot. Avant de charger l'algorithme de détection, il est nécessaire de définir les trois paramètres suivants :

1. L'angle α.
2. Les vitesses angulaire et linéaire maximale.
3. La navigation semi-autonome (priorité de commande).

C.2.6 Choix de l'angle α

Le principe de détection des obstacles négatifs repose sur la classification de variations d'angle d'appareil laser allant de 0° jusqu' à 180° ayant un rayon qui varie entre $R = 180$cm et $R + \Delta R$=220cm(cas d'une dénivellation de 10 cm). Le rayon R (rayon de référence)est expérimentalement choisi de telle sorte que l'angle de balayage soit à 89°. On remarque que plus l'angle α est grand, plus la distance est petite. Cependant en pratique, il apparait de nouvelles contraintes en l'occurrence relatives aux virages et la détection des obstacles positives.

Les critères pour choisir α sont :

$$R \leq R1$$

Cette première condition est nécessaire pour que le robot puisse effectuer un virage tout en respectant les critères de sécurité imposés par l'usager soit une zone de navigation permise. À titre d'exemple, on peut imaginer que le robot utilisé soit une chaise roulante. Toujours dans le cadre d'une manoeuvre de tournant, il est donné la possibilité à l'usager d'utiliser le joystick moyennant

la méthode dite d'avancement des points expliquée ultérieurement. Cette méthode pourvoit l'usager d'une garantie de manoeuvre plus sécuritaire.

$$R \geq R2$$

Où $R2$ est la distance minimale à partir de laquelle le robot doit prendre une décision réactive eu égard à une détection d'obstacles à proximité Le rayon R est le rayon de référence pris avec un angle de 89°. La Figure C.3 illustre un cas de virage et met en évidence les deux critères à respecter lors du choix de α.

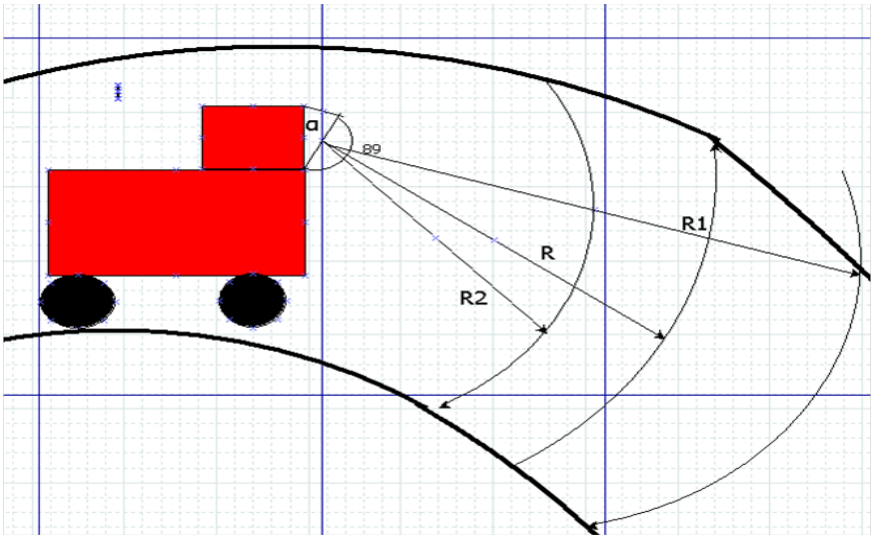


Figure C.3 Les critères pour choisir α

C.2.7 Choix de vitesses angulaire et linéaire

La vitesse linéaire est choisie en considérant les contraintes liées à l'environnement et aux contraintes holonomiques du robot. Ce faisant, la vitesse maximale devra respecter ces contraintes. Le NavPoinToPoint choisit sa vitesse en considérant le trajet et sa forme (longueur de chaque segment) et tient compte du confort de l'usager.

Pour la vitesse angulaire, sa valeur maximale est choisie en considérant exclusivement les contraintes liées à l'environnement. Cela est d'autant plus vrai qu'on navigue sur un trottoir limité par une ou deux dénivellations. Le NavPoinToPoint choisi la vitesse suite à la détection d'un obstacle. Cependant dans le cas où l'obstacle est à une distance critique, il est laissé la liberté au mode réactif de choisir une vitesse angulaire appropriée.

C.3 Navigation semi-autonome

La navigation à l'extérieur nécessite que les manoeuvres soient effectuées prudemment puisque il s'agit d'un environnement impliquant diverses situations inconnues. D'ailleurs cela oblige l'usager à opérer souvent en mode manuel ou semi autonome du fait de l'impossibilité de prévoir toutes les situations probables. Lorsqu'il est question d'une décision à prendre par l'usager impliquant l'utilisation du joystick, le NavPoinToPoint vient assister la manoeuvre à effectuer. Cela se produit en chargeant les points nouvellement sélectionnés par l'action sur le Joystick, dans la liste de NavPoinToPoint permettant par delà au robot de continuer à avancer en satisfaisant la condition de passage par les cercles de la trajectoire suivie.

Ainsi, à chaque fois que l'usager touche le joystick pour prendre des décisions et appliquer une commande directe, le NavPoinToPoint commence à calculer la distance parcourue pour déterminer le point de ralliement à la liste, le plus proche de l'endroit où se trouve le robot actuellement. Après avoir calculé la distance parcourue, il faut déterminer le nombre de points outrepassé durant le mode manuel, correspondants à cette distance et décaler ces derniers dans la liste pour trouver le point de ralliement à la trajectoire préétablie par l'algorithme NavPoinToPoint. Ce type de commande est dit collaboratif car il accorde par alternance le contrôle à l'usager ainsi qu'à un algorithme spécial destiné à réaliser une commande optimale.

En fait dans la dernière partie du circuit Acropolis, il y a un plugin qui émet des commandes pour contrôler le moteur. Dans ce plugin la commande collaborative dite commande mixte est employée pour déterminer les vitesses angulaire et linéaire. À titre d'exemple, citons le cas d'une manoeuvre d'évitement d'obstacle, dans laquelle la commande de vitesse linéaire est produite par l'algorithme NavPoinToPoint alors que la commande de vitesse angulaire provient elle, du mode réactif. Le changement de mode (manuel, semi-autonome,...) est nécessaire pour mieux contrôler le robot, Par exemple dans le cas d'un échec de l'algorithme pour caractériser une situation rencontrée suite à un manque voire une absence de données nécessaires à cela (caméra, inclinomètre,...) il est donné à l'usager la possibilité de choisir manuellement son propre chemin.

Le nombre de points d'avancement dans le NavPoinToPoint choisi est égal à 3 points pour chaque touché du Joystick. Il est à noter que le capteur d'obstacle, soit le laser, fonctionne selon un cycle de scan de $100\mu s$. Ainsi les données à analyser étant importantes ; le calcul devient fastidieux et le temps de réaction altéré. Dès lors le défi consiste à sélectionner les données les plus pertinentes qui aideraient à prendre la décision la plus rapide et la plus fiable en termes de sécurité. Aussi, il faut relever le fait que la conception de tous les algorithmes du circuit doit tenir compte de cette contrainte temps de calcul autrement faute de réaction à temps les situations de collision entre le robot et les obstacles seront plus de nombreuses.

La figure C.4 illustre la réaction du robot lorsque ce dernier détecte un obstacle soit une dévia-

tion de sa trajectoire à droite ou à gauche de l'obstacle en question. Une fois que l'usager lâche le joystick, le plugin PlanAlgo génère de nouveaux points pour garantir au robot un contournement de l'obstacle de manière sécuritaire.

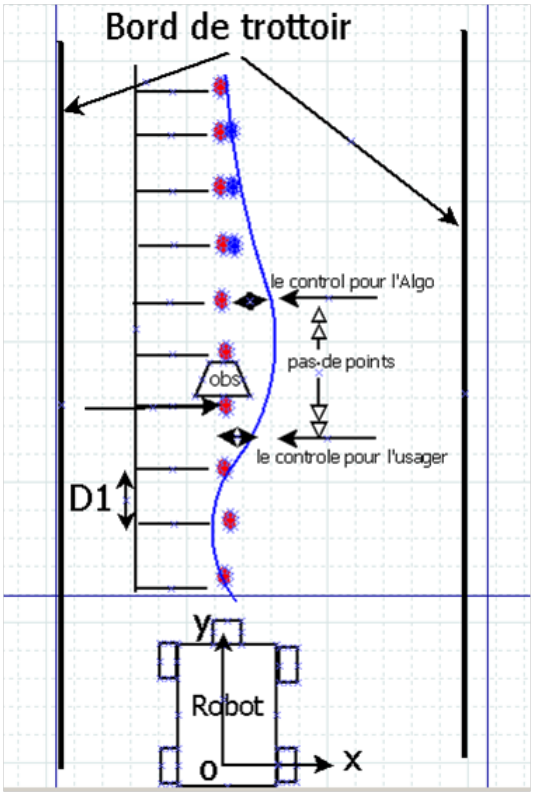


Figure C.4 Évitement des obstacles en planifiant les points à l'avance

C.4 NavPoinToPoint

Étant donné que le robot mobile a des contraintes holonomes et non holonomes (il y a plus d'action de contrôle que de variable de contrôle) qui l'empêchent d'atteindre un point dans le plan *xoy*, alors il faut créer un module pour naviguer dans ce plan *xoy*. En partant de ces contraintes ; il est indispensable de trouver une moyenne de navigation dans l'espace $2D$.

Cette moyenne est caractérisée par un plugin, dans l'environnement Acropolis, appelé NavPointToPoint. Ce dernier est créé pour un type de déplacement en $2D$ utilisant cependant une carte pour aider le robot à se déplacer et se localiser. Il choisit les points sécuritaires et effectue des évitements d'obstacles réactifs. Dans un environnement restreint la navigation planifiée est souvent interrompue au bénéfice d'une procédure de réaction face à des situations imprévues. En effet, les points parcourus, choisis à priori, ne dépassent pas 4 où 5 au maximum (cas d'un obstacle). L'ex-

périence à montrer que ce plugin associé avec notre Algorithme, nous confère plusieurs avantages mais aussi certains inconvénients. Quelques un d'entre eux sont cités ci-dessous

C.4.1 Avantages

1. Organisation des points à naviguer dans une liste. Les points non listés sont automatiquement assignés à un élément de la liste dès que le robot se trouve dans un rayon préalablement défini autour de cet élément. La figure C.5 montre les points et leurs rayons correspondants.
2. Choix des vitesses angulaire et linéaire de telle sorte que ces dernières satisfassent des critères de confort pour l'usager (pas de déplacement brusque).
3. Pas de planification de trajectoire en ligne dans la procédure d'évitement d'obstacles se trouvant devant le robot car cette planification ne peut être faite que dans les zones libres de la carte.
4. Utilisation de coefficients de configuration (les vitesses angulaire et linéaire maximal, le rayon des points de la liste, l'erreur du point d'arrivée, les pôles pour la cinématique inversée etc.) donnant à ce plugin la capacité de parcourir le trajet de manière optimal en termes de contraintes établies à priori (confort usager, sécurité etc.).

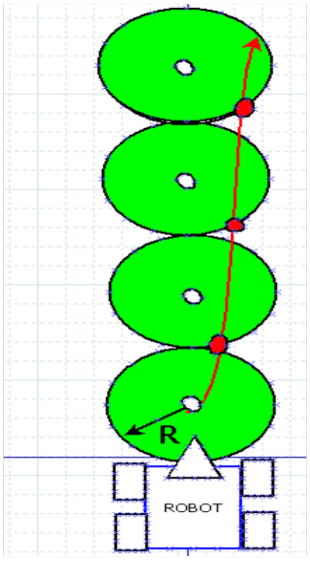


Figure C.5 Les points d'un trajet pour le plugin NavPointToPoint

C.4.2 Inconvénients

1. Le nombre de point constituant la liste dépasse 4 bien que l'environnement de travail soit restreint.

2. Valide pour un court et approximatif chemin entre les points.
3. De nouveaux points sont pris dès que le robot arrive au rayon *R* entourant les points de ralliement de la trajectoire planifiée et cela est considéré comme un inconvénient dans le design, pourquoi ? Parce que les erreurs accumulées par l'odométrie lors d'un long trajet s'ajoutent à la liste des points de NavPointToPoint (sans carte). Autrement dit, le robot croit qu'il a atteint le dernier point dès qu'il touche le rayon du point de ralliement. Par conséquent, cela engendre une erreur accumulée avec le temps dûe à la méthode de conception du plugin. La figure suivante explique pourquoi le NavPointToPoint n'est pas un choix idéal pour la navigation longue distance, sans carte.
4. Pas de planification de trajectoire en ligne dans la procédure d'évitement d'obstacles se trouvant devant le robot car cette planification ne peut être faite que dans les zones libres de la carte.

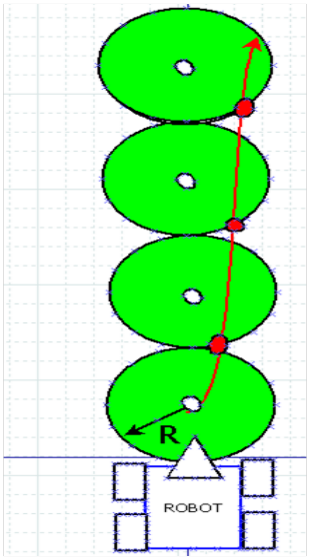


Figure C.6 Les points d'un trajet pour le plugin NavPointToPoint

En énumérant les avantages et inconvénients, on en déduit que des modifications doivent être apportées à la façon dont le plugin NavPointToPoint choisit ses points. Ces modifications doivent permettre à la navigation d'éviter le plus souvent d'avoir recours au mode réactif en planifiant une trajectoire d'évitement dès la détection d'un obstacle.

La solution consiste à ajouter une valeur fixe à chaque point de la liste pour que le robot puisse atteindre la deuxième partie du cercle comme illustré sur la figure ci-dessous. Ce faisant, il faut aussi rapprocher les points de la liste ; cela corrige les erreurs d'odométrie et garantie la continuité de la navigation longue distance.

Durant la navigation, les points donnés dépassent la première partie de cercle A. Le robot s'arrête à la partie colorée ce qui garantit le passage de la trajectoire à proximité des centres des cercles. La procédure se répète ainsi pour chaque point de la liste. Les points à considérer dans la liste sont les points A', B', C', D' à la place des points A,B,C,D.

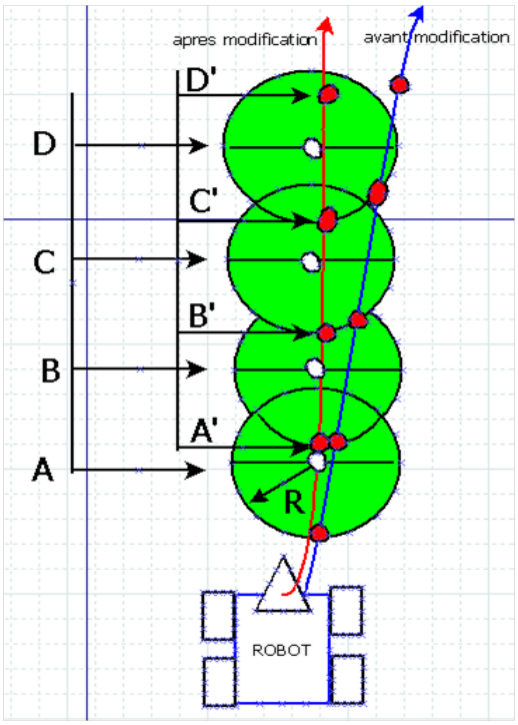


Figure C.7 Chemin du Plugin NavPointToPoint et le chemin modifié

C.4.3 Cas d'un obstacle éloigné

Dans le cas de la détection d'un obstacle éloigné, le robot doit pouvoir planifier une nouvelle trajectoire auxiliaire qui permet à ce dernier de s'écarter soit à droite soit à gauche selon la position et les dimensions de l'obstacle à éviter. Cette manoeuvre suscite une génération de points avant l'obstacle (2 points) et après lui (2 points). La déviation est estimée par l'écart de distance entre l'obstacle et le robot ainsi que par la distance séparant le robot des bords du trottoir. Subséquemment, on aurait une trajectoire planifiée dévolue à l'évitement d'obstacles n'ayant pas recours au mode réactif entrainant la plupart du temps des actions inconfortables et non sécuritaires. La figure ci-après montre un chemin alternatif dans le cas de présence d'un obstacle éloigné.

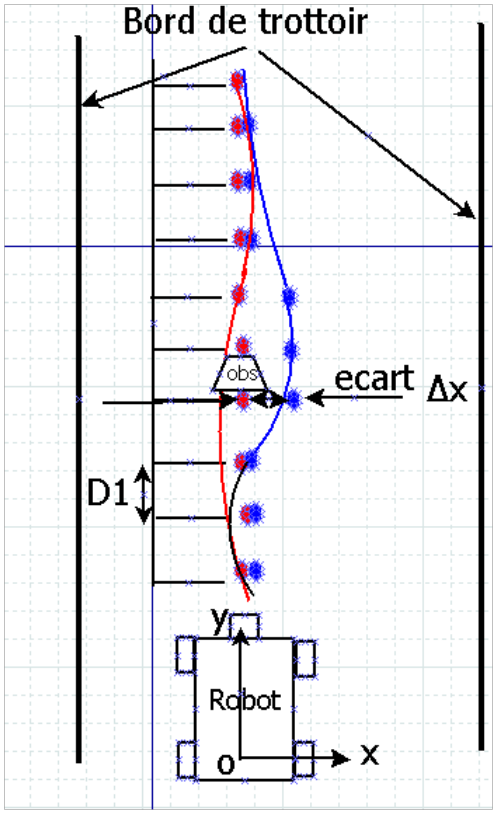


Figure C.8 Évitement d'un obstacle éloigné

C.4.4 Cas d'un obstacle proche

La détection des obstacles proches par un laser incliné est une tâche difficile. De plus, pour éviter ces obstacles (poubelle, colonne d'électricité, arbre, etc.) le mode réactif est utilisé dans le plugin qui émet les commandes directement au moteur soit le plugin Joymanuel. Les obstacles proches sont caractérisés par une distance plus proche que celle des points de contact avec le trottoir. En effet, il faut appliquer une commande directe sur le moteur : la vitesse angulaire doit être changée pour contourner l'obstacle ou bien faire dévier le robot le plus loin possible de ce dernier. La commande appliquée doit prendre en considération les contraintes imposées par l'environnement et les bordures de trottoir (limites).

C.4.5 Détection des dénivellations

La détection des dénivellations emploie le plugin Joymanuel (qui émet des commandes directes au moteur) afin de détecter les dénivellations, les trous et les obstacles puis les éviter. L'algorithme cité en sus est organisé tel suit :

1. détection des dénivellations $Den1$et $Den2$.
2. détection des obstacles proches.

Les décisions émises sont organisées comme suit :

1. décision pour la vitesse angulaire basée sur la détection des $Den1$ et $Den2$.
2. décision pour la vitesse angulaire basée sur la détection d'obstacles proches : cette commande est directe et non discutable, contrairement au cas précédent de la détection des $Den1$ et $Den2$ qui conduit à une décision vers une boucle de commandes directes.

C.4.6 Détection des dénivellations Den1 et Den2 et la commande correspondante

Cette détection est basée sur l'idée d'estimer la position angulaire à partir de la comparaison de deux niveaux de surfaces différents (surface de progression du robot soit le trottoir et surface en dessous de cette dernière soit la route). La méthode intuitive consiste à scanner toute la région visible cependant en pratique le traitement pour chaque segment étant différent, cela conduirait à un calcul fastidieux avec des résultats non livrés à temps. Par conséquent, il sera plutôt considéré une zone d'intérêt entre 60 et 120 degrés garantissant une navigation sécuritaire. Au début, il y a 4 scénarios possibles :

Le premier scenario est l'existence de deux bords de trottoir caractérisés par deux dénivellations $Den1$ **et** $Den2$:

Le robot navigue sur un trottoir avec deux bords. Le plugin Joymanuel détermine $Den1$ et $Den2$ grâce au laser qui transmet l'information relative à la distance critique au delà de laquelle le niveau

de la plateforme où évolue le robot change. Si le robot, est au milieu du terrain avec une marge de distance vis-à-vis de chaque bord ; ce dernier garde ces marges de distance respectivement à gauche (*MRDG* : marge de distance à gauche) et à droite (*MRDD* : marge de distance à droit). Les limites permises, pour que le robot reste dans une position sécuritaire, sont déterminées dépendamment de la largeur du trottoir. Ainsi, si le trottoir est très étroit, alors les *MRDG* et *MRDD* seront très petits soit entre 5cm et 10 cm. Pour que la navigation semi-autonome s'effectue de manière sécuritaire dans un environnement restreint les *MRDG* et *MRDD* doivent donc être respectées. La commande émise aura pour objectif de faire tourner le robot respectivement à gauche si le seuil *MRDD* est moins de ce qui est permis, et à droite si le seuil *MRDG* est franchi. Le robot ne change pas la vitesse angulaire si les *MRDG* et *MRDD* sont dans les limites permises. La vitesse linéaire quant à elle reste fixe à moins qu'il y ait un obstacle devant le robot.

Le robot navigue sur un trottoir avec deux bords. Le plugin Joymanuel détermine *Den*1 et *Den*2 grâce au laser qui transmet l'information relative à la distance critique au delà de laquelle le niveau de la plateforme où évolue le robot change. Si le robot, est au milieu du terrain avec une marge de distance vis-à-vis de chaque bord ; ce dernier garde ces marges de distance respectivement à gauche (*MRDG* : marge de distance à gauche) et à droite (*MRDD* : marge de distance à droit). Les limites permises, pour que le robot reste dans une position sécuritaire, sont déterminées dépendamment de la largeur du trottoir. Ainsi, si le trottoir est très étroit, alors les *MRDG* et *MRDD* seront très petits soit entre 5*cm* et 10 cm. Pour que la navigation semi-autonome s'effectue de manière sécuritaire dans un environnement restreint les *MRDG* et *MRDD* doivent donc être respectées. La commande émise aura pour objectif de faire tourner le robot respectivement à gauche si le seuil *MRDD* est moins de ce qui est permis, et à droite si le seuil *MRDG* est franchi. Le robot ne change pas la vitesse angulaire si les *MRDG* et *MRDD* sont dans les limites permises. La vitesse linéaire quant à elle reste fixe à moins qu'il y ait un obstacle devant le robot.

Le deuxième scenario est l'absence de *Den*2 :

Dans ce cas le Joymanuel doit estimer le *Den*2 car l'algorithme nécessite la manipulation de deux paramètres soit *Den*1 et *Den*2. Pour ce faire il faut prendre en considération le *Den*1. Ce paramètre est calculé comme mentionné dans la section précédente ; il ne reste alors plus qu'à estimer *Den*2. L'estimation de *Den*2 dépend de la position du robot sur le trottoir spécifiant la condition suivante : 'le robot peut il naviguer en toute sécurité du coté de *Den*2'. Ainsi, le laser scanne la région à hauteur de 30° (angle empirique défini expérimentalement) ajouté à l'angle avec lequel le laser détecte *Den*1. Dépendamment de si le robot détecte ou non ce deuxième bord, le Joymanuel estime une position pour *Den*2 qui garantirait une valeur permettant au robot de naviguer dans les limites imposées par les *MRDG* et *MRDD*. Cette procédure a été testée et a donné de bons résultats. L'absence de *Den*2 vient du fait que le trottoir a juste un côté droit, l'autre côté étant soit un terrain soit un mur etc. L'important est de ne pas franchir les limites de *Den*1 et *Den*2.

Le troisième scenario est l'absence de *Den*1 :

L'absence de *Den*1 peut être due aux mêmes conjectures citées dans le scénario précédent. Dans ce cas, le laser fait un scan de 0° à 89°. Si la position de *Den*1 n'est pas détectée, une valeur initiale aléatoire lui sera attribuée. Toutefois dès la détection de *Den*2, la valeur de *Den*1 sera actualisé ce qui assure au robot une navigation continue. cette méthode considère le cas de la navigation dans un environnement restreint qui contraint le robot à n'effectuer que des manoeuvres dans le cadre des limitations prescrites. Les résultats expérimentaux sur le trottoir étaient concluants en ce sens que le robot a fait une manouvre pour garder la *MRDG* dans la limite permise. Quant aux commandes émises. Ces dernières sont issues et de l'algorithme d'évitement de la dénivellation *Den*2 et des commande réactives pour l'évitement d'autres obstacles.

Le quatrième scenario est l'absence de *Den*1 **et** *Den*2 :

Considérant un terrain homogène, donc pas de dénivellations à considérer ; il faut adapter l'algorithme du laser incliné en attribuant des valeurs aux *Den*1, *Den*2 et en continuant la navigation. Dans ce type de navigation les limites *MRDG* et *MRDD* sont en vérité toujours respectées. Pour les commandes émises le NavPointToPoint gère la navigation et transmet les commandes appropriées comme s'il s'agissait d'une navigation sans embuches.

C.4.7 Alignement avec un mur

Le principe de s'aligner avec un mur est de garder une distance fixe dans laquelle une marge a été met pour donner le robot plus de flexibilité lors de rapproche du mur. Cette distance peut être déterminée par les dimensions physiques de robot et les dimensions de trottoir où le robot fait son parcour. Dans la Figure C.9, on remarque qu' la ligne rouge est la ligne après lequel aucun mouvement n'est permis. La zone du danger a été déterminée par la ligne rouge et le mur.

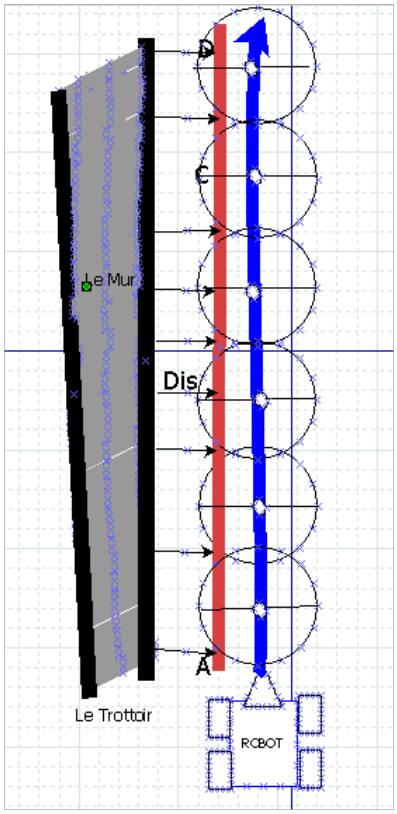


Figure C.9 Évitement d’un obstacle éloigné

C.5 Conclusion

Dans ce chapitre, on a présenté une méthode pour détecter les obstacles et les dénivellations avec un laser incliné. Les résultats attendus étaient assez acceptable pour une navigation prudente. La définition des obstacles peut varier selon l'appareil utilisé et le type des obstacles à détecter. Les obstacles sont divisés en deux catégories : négatif, positif. L'angle de l'inclinaison a été choisi selon des critères liés au chemin à naviguer. La planification est nécessaire pour faire un évitement rapide, sécuritaire et confortable ; c'est pour cela on a fait l'évitement à deux niveaux. Dans ce cadre on a utilisé un plugin de contrôle (NavPointToPoint) dans AcroPolis pour faire ces manouvres. À l'annexe B, des modifications sont rapportées à ce plugin pour qu'il fasse le travail demandé avec efficacité.